Oracle® Application Express

API Reference

Release 4.0

E15519-01

June 2010

ORACLE®

Oracle Application Express API Reference, Release 4.0

E15519-01

Copyright © 2003, 2010, Oracle and/or its affiliates. All rights reserved.

Primary Author: Drue Baker

Contributors: Marco Adelfio, Drue Baker, Carl Backstrom, Christina Cho, Steve Fogel, Michael Hichwa, Terri Jennings, Christopher Jones, Joel Kallman, Sharon Kennedy, Syme Kutz, Sergio Leunissen, Anne Romano, Kris Rice, Marc Sewtz, Scott Spadafore, Scott Spendolini, Jason Straub, and Simon Watt.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface xiii

Topic Overview xiii
Audience xiv
Documentation Accessibility xiv
Related Documents xv
Conventions xvi

What's New in this Document xvii

Oracle Application Express API Reference Updates for Release 4.0 xvii

1 APEX_UTIL

CACHE_GET_DATE_OF_PAGE_CACHE Function 1-5
CACHE_GET_DATE_OF_REGION_CACHE Function 1-6
CACHE_PURGE_BY_APPLICATION Procedure 1-7
CACHE_PURGE_BY_PAGE Procedure 1-8
CACHE_PURGE_STALE Procedure 1-9
CHANGE_CURRENT_USER_PW Procedure 1-10
CHANGE_PASSWORD_ON_FIRST_USE Function 1-11
CLEAR_APP_CACHE Procedure 1-12
CLEAR_PAGE_CACHE Procedure 1-13
CLEAR_USER_CACHE Procedure 1-14
COUNT_CLICK Procedure 1-15
CREATE_USER Procedure 1-16
CREATE_USER_GROUP Procedure 1-19
CURRENT_USER_IN_GROUP Function 1-20
DOWNLOAD_PRINT_DOCUMENT Procedure Signature 1 1-21
DOWNLOAD_PRINT_DOCUMENT Procedure Signature 2 1-22
DOWNLOAD_PRINT_DOCUMENT Procedure Signature 3 1-24
DOWNLOAD_PRINT_DOCUMENT Procedure Signature 4 1-26
EDIT_USER Procedure 1-27
END_USER_ACCOUNT_DAYS_LEFT Function 1-31
EXPIRE_END_USER_ACCOUNT Procedure 1-32

EXPIRE_WORKSPACE_ACCOUNT Procedure 1-33
EXPORT_USERS Procedure 1-34
FETCH_APP_ITEM Function 1-35
FETCH_USER Procedure Signature 1 1-36
FETCH_USER Procedure Signature 2 1-39
FETCH_USER Procedure Signature 3 1-41
FIND_SECURITY_GROUP_ID Function 1-44
FIND_WORKSPACE Function 1-45
GET_ACCOUNT_LOCKED_STATUS Function 1-46
GET_ATTRIBUTE Function 1-47
GET_AUTHENTICATION_RESULT Function 1-48
GET_BLOB_FILE_SRC Function 1-49
GET_CURRENT_USER_ID Function 1-51
GET_DEFAULT_SCHEMA Function 1-52
GET_EDITION Function 1-53
GET_EMAIL Function 1-54
GET_FEEDBACK_FOLLOW_UP Function 1-55
GET_FILE Procedure 1-56
GET_FILE_ID Function 1-58
GET_FIRST_NAME Function 1-59
GET_GROUPS_USER_BELONGS_TO Function 1-60
GET_GROUP_ID Function 1-61
GET_GROUP_NAME Function 1-62
GET_LAST_NAME Function 1-63
GET_NUMERIC_SESSION_STATE Function 1-64
GET_PREFERENCE Function 1-65
GET_PRINT_DOCUMENT Function Signature 1 1-66
GET_PRINT_DOCUMENT Function Signature 2 1-67
GET_PRINT_DOCUMENT Function Signature 3 1-68
GET_PRINT_DOCUMENT Function Signature 4 1-69
GET_SCREEN_READER_MODE_TOGGLE Function 1-71
GET_SESSION_LANG Function 1-72
GET_SESSION_STATE Function 1-73
GET_SESSION_TERRITORY Function 1-74
GET_SESSION_TIME_ZONE Function 1-75
GET_USER_ID Function 1-76
GET_USER_ROLES Function 1-77
GET_USERNAME Function 1-78
IR_CLEAR Procedure 1-79
IR_FILTER Procedure 1-80
IR_RESET Procedure 1-81
IS_LOGIN_PASSWORD_VALID Function 1-82

IS_SCREEN_READER_SESSION Function 1-83
IS_SCREEN_READER_SESSION_YN Function 1-84
IS_USERNAME_UNIQUE Function 1-85
KEYVAL_NUM Function 1-86
KEYVAL_VC2 Function 1-87
LOCK_ACCOUNT Procedure 1-88
PASSWORD_FIRST_USE_OCCURRED Function 1-89
PREPARE_URL Function 1-90
PUBLIC_CHECK_AUTHORIZATION Function 1-92
PURGE_REGIONS_BY_APP Procedure 1-93
PURGE_REGIONS_BY_NAME Procedure 1-94
PURGE_REGIONS_BY_PAGE Procedure 1-95
REMOVE_PREFERENCE Procedure 1-96
REMOVE_SORT_PREFERENCES Procedure 1-97
REMOVE_USER Procedure 1-98
RESET_AUTHORIZATIONS Procedure 1-99
RESET_PW Procedure 1-100
SAVEKEY_NUM Function 1-101
SAVEKEY_VC2 Function 1-102
SET_ATTRIBUTE Procedure 1-103
SET_AUTHENTICATION_RESULT Procedure 1-104
SET_CUSTOM_AUTH_STATUS Procedure 1-105
SET_EDITION Procedure 1-106
SET_EMAIL Procedure 1-107
SET_FIRST_NAME Procedure 1-108
SET_LAST_NAME Procedure 1-109
SET_PREFERENCE Procedure 1-110
SET_SECURITY_GROUP_ID Procedure 1-111
SET_SESSION_LANG Procedure 1-112
SET_SESSION_LIFETIME_SECONDS Procedure 1-113
SET_SESSION_MAX_IDLE_SECONDS Procedure 1-115
SET_SESSION_SCREEN_READER_OFF Procedure 1-117
SET_SESSION_SCREEN_READER_ON Procedure 1-118
SET_SESSION_STATE Procedure 1-119
SET_SESSION_TERRITORY Procedure 1-120
SET_SESSION_TIME_ZONE Procedure 1-121
SET_USERNAME Procedure 1-122
SHOW_SCREEN_READER_MODE_TOGGLE Procedure 1-123
STRING_TO_TABLE Function 1-124
STRONG_PASSWORD_CHECK Procedure 1-125
STRONG_PASSWORD_VALIDATION Function 1-129

SUBMIT_FEEDBACK Procedure 1-130
SUBMIT_FEEDBACK_FOLLOWUP Procedure 1-132
TABLE_TO_STRING Function 1-133
UNEXPIRE_END_USER_ACCOUNT Procedure 1-134
UNEXPIRE_WORKSPACE_ACCOUNT Procedure 1-135
UNLOCK_ACCOUNT Procedure 1-136
URL_ENCODE Function 1-137
WORKSPACE_ACCOUNT_DAYS_LEFT Function 1-139

2 APEX_MAIL

About Configuring Oracle Application Express to Send Email 2-2
ADD_ATTACHMENT Procedure 2-3
PUSH_QUEUE Procedure 2-4
SEND Procedure 2-5

3 APEX_ITEM

CHECKBOX Function 3-2
DATE_POPUP Function 3-4
DISPLAY_AND_SAVE Function 3-6
HIDDEN Function 3-7
MD5_CHECKSUM Function 3-9
MD5_HIDDEN Function 3-10
POPUP_FROM_LOV Function 3-11
POPUP_FROM_QUERY Function 3-13
POPUPKEY_FROM_LOV Function 3-15
POPUPKEY_FROM_QUERY Function 3-17
RADIOGROUP Function 3-19
SELECT_LIST Function 3-20
SELECT_LIST_FROM_LOV Function 3-22
SELECT_LIST_FROM_LOV_XL Function 3-23
SELECT_LIST_FROM_QUERY Function 3-25
SELECT_LIST_FROM_QUERY_XL Function 3-27
TEXT Function 3-29
TEXTAREA Function 3-30
TEXT_FROM_LOV Function 3-31
TEXT_FROM_LOV_QUERY Function 3-32

4 APEX_APPLICATION

Referencing Arrays 4-3
Referencing Values Within an On Submit Process 4-4
Converting an Array to a Single Value 4-5
HELP Procedure 4-6

5 APEX_CUSTOM_AUTH

APPLICATION_PAGE_ITEM_EXISTS Function 5-2
CURRENT_PAGE_IS_PUBLIC Function 5-3
DEFINE_USER_SESSION Procedure 5-4
GET_COOKIE_PROPS Procedure 5-5
GET_LDAP_PROPS Procedure 5-6
GET_NEXT_SESSION_ID Function 5-8
GET_SECURITY_GROUP_ID Function 5-9
GET_SESSION_ID Function 5-10
GET_SESSION_ID_FROM_COOKIE Function 5-11
GET_USER Function 5-12
GET_USERNAME Function 5-13
IS_SESSION_VALID Function 5-14
LOGIN Procedure 5-15
LOGOUT Procedure 5-16
POST_LOGIN Procedure 5-17
SESSION_ID_EXISTS Function 5-18
SET_SESSION_ID Procedure 5-19
SET_SESSION_ID_TO_NEXT_VALUE Procedure 5-20
SET_USER Procedure 5-21

6 APEX_LDAP

AUTHENTICATE Function 6-2
GET_ALL_USER_ATTRIBUTES Procedure 6-3
GET_USER_ATTRIBUTES Procedure 6-5
IS_MEMBER Function 6-7
MEMBER_OF Function 6-9
MEMBER_OF2 Function 6-10

7 APEX_INSTANCE_ADMIN

ADD_SCHEMA Procedure 7-2
ADD_WORKSPACE Procedure 7-3
GET_PARAMETER Function 7-4
GET_SCHEMAS Function 7-5
REMOVE_SAVED_REPORT Procedure 7-6
REMOVE_SAVED_REPORTS Procedure 7-7
REMOVE_SCHEMA Procedure 7-8
REMOVE_WORKSPACE Procedure 7-9
SET_PARAMETER Procedure 7-10
Available Parameter Values 7-11

8 APEX_UI_DEFAULT_UPDATE

ADD_AD_COLUMN Procedure 8-3
ADD_AD_SYNONYM Procedure 8-5
DEL_AD_COLUMN Procedure 8-6
DEL_AD_SYNONYM Procedure 8-7
DEL_COLUMN Procedure 8-8
DEL_GROUP Procedure 8-9
DEL_TABLE Procedure 8-10
SYNCH_TABLE Procedure 8-11
UPD_AD_COLUMN Procedure 8-12
UPD_AD_SYNONYM Procedure 8-14
UPD_COLUMN Procedure 8-15
UPD_DISPLAY_IN_FORM Procedure 8-17
UPD_DISPLAY_IN_REPORT Procedure 8-18
UPD_FORM_REGION_TITLE Procedure 8-19
UPD_GROUP Procedure 8-20
UPD_ITEM_DISPLAY_HEIGHT Procedure 8-21
UPD_ITEM_DISPLAY_WIDTH Procedure 8-22
UPD_ITEM_FORMAT_MASK Procedure 8-23
UPD_ITEM_HELP Procedure 8-24
UPD_LABEL Procedure 8-25
UPD_REPORT_ALIGNMENT Procedure 8-26
UPD_REPORT_FORMAT_MASK Procedure 8-27
UPD_REPORT_REGION_TITLE Procedure 8-28
UPD_TABLE Procedure 8-29

9 JavaScript APIs

apex.event.trigger(pSelector,pEvent,pData) 9-4
apex.widget.initPageItem(pName,pOptions) 9-5
$x(pNd) 9-6
$v(pNd) 9-7
$v2(pNd) 9-8
$s(pNd, pValue) 9-9
$u_Carray(pNd) 9-10
$u_Narray(pNd) 9-11
$nvl(pTest, pDefault) 9-12
apex.submit(pRequest | pOptions) 9-13
apex.confirm(pMessage, pRequest) 9-14
apex.confirm(pMessage, pOptions) 9-15
$x_Style(pNd, pStyle, pString) 9-16
$x_Hide(pNd) 9-17

$x_Show(pNd) 9-18
$x_Toggle(pNd) 9-19
$x_Remove(pNd) 9-20
$x_Value(pNd,pValue) 9-21
$x_UpTill(pNd, pToTag) 9-22
$x_ItemRow(pNd,pFunc) 9-23
$x_HideItemRow(pNd) 9-24
$x_ShowItemRow(pNd) 9-25
$x_ToggleItemRow(pNd) 9-26
$x_HideAllExcept(pNd,pNdArray) 9-27
$x_HideSiblings(pNd) 9-28
$x_ShowSiblings(pNd) 9-29
$x_Class(pNd,pClass) 9-30
$x_SetSiblingsClass(pNd, pClass, pNdClass) 9-31
$x_ByClass(pClass, pNd, pTag) 9-32
$x_ShowAllByClass(pNd, pClass, pTag) 9-33
$x_ShowChildren(pNd) 9-34
$x_HideChildren(pNd) 9-35
$x_disableItem(pNd, pTest) 9-36
$f_get_emptys(pNd, pClassFail, pClass) 9-37
$v_Array(pNd) 9-38
$f_ReturnChecked(pNd) 9-39
$d_ClearAndHide(pNd) 9-40
$f_SelectedOptions(pNd) 9-41
$f_SelectValue(pNd) 9-42
$u_ArrayToString(pArray, pDelim) 9-43
$x_CheckImageSrc(pId,pSearch) 9-44
$v_CheckValueAgainst(pThis, pValue) 9-45
$f_Hide_On_Value_Item(pThis, pThat, pValue) 9-46
$f_Show_On_Value_Item(pThis, pThat, pValue) 9-47
$f_Hide_On_Value_Item_Row(pThis, pThat, pValue) 9-48
$f_Show_On_Value_Item_Row(pThis, pThat, pValue) 9-49
$f_DisableOnValue(pThis, pValue, pThat) 9-50
$x_ClassByClass(pNd, pClass, pTag, pClass2) 9-51
$f_ValuesToArray(pThis, pClass, pTag) 9-52
$x_FormItems(pNd, pType) 9-53
$f_CheckAll(pThis, pCheck, pArray) 9-54
$f_CheckFirstColumn(pNd) 9-55
$v_PopupReturn(pValue, pThat) 9-56
$x_ToggleWithImage(pThis,pNd) 9-57
$x_SwitchImageSrc(pNd, pSearch, pReplace) 9-58

$x_CheckImageSrc(pNd, pSearch) ... 9-59
$u_SubString(pText,pMatch) ... 9-60
html_RemoveAllChildren(pNd) ... 9-61
$v_IsEmpty(pThis) ... 9-62
html_SetSelectValue(pId,pValue) ... 9-63
addLoadEvent(pFunction) ... 9-64
$f_Swap(pThis,pThat) ... 9-65
submitEnter(pNd,e) ... 9-66
$f_SetValueSequence(pArray,pMultiple) ... 9-67
$dom_AddTag(pThis, pTag, pText) ... 9-68
$tr_AddTD(pThis,pText) ... 9-69
$tr_AddTH(pThis,pText) ... 9-70
$dom_AddInput(pThis,pType,pId,pName,pValue) ... 9-71
$dom_MakeParent(p_Node,p_Parent) ... 9-72
$x_RowHighlight(pThis, pColor) ... 9-73
$x_RowHighlightOff(pThis) ... 9-74
$v_Upper(pNd) ... 9-75
$d_Find(pThis,pString,pTags,pClass) ... 9-76
returnInput(p_R, p_D) ... 9-77
setReturn(p_R,p_D) ... 9-78
$f_First_field(pNd) ... 9-79
GetCookie (pName) ... 9-80
SetCookie (pName,pValue) ... 9-81

10 APEX_PLSQL_JOB

About the APEX_PLSQL_JOB Package ... 10-2
JOBS_ARE_ENABLED Function ... 10-3
PURGE_PROCESS Procedure ... 10-4
SUBMIT_PROCESS Function ... 10-5
TIME_ELAPSED Function ... 10-6
UPDATE_JOB_STATUS Procedure ... 10-7

11 APEX_LANG

LANG Function ... 11-2
MESSAGE Function ... 11-3

12 APEX_COLLECTION

About the APEX_COLLECTION API ... 12-3
Naming, Creating and Accessing Collections ... 12-4
Naming Collections ... 12-4
Creating a Collection ... 12-4
About the Parameter p_generate_md5 ... 12-5

Accessing a Collection 12-5
Merging, Truncating and Deleting Collections 12-7
Merging Collections 12-7
Truncating a Collection 12-7
Deleting a Collection 12-7
Deleting All Collections for the Current Application 12-7
Deleting All Collections in the Current Session 12-7
Adding, Updating and Deleting Collection Members 12-8
Adding Members to a Collection 12-8
About the Parameters p_generate_md5, p_clob001, p_blob001, and p_xmltype001 12-8
Updating Collection Members 12-8
Deleting Collection Members 12-9
Managing Collections 12-10
Obtaining a Member Count 12-10
Resequencing a Collection 12-10
Verifying Whether a Collection Exists 12-10
Adjusting a Member Sequence ID 12-10
Sorting Collection Members 12-10
Clearing Collection Session State 12-10
Determining Collection Status 12-11
ADD_MEMBER Procedure 12-12
ADD_MEMBER Function 12-14
ADD_MEMBERS Procedure 12-16
COLLECTION_EXISTS Function 12-18
COLLECTION_HAS_CHANGED Function 12-19
COLLECTION_MEMBER_COUNT Function 12-20
CREATE_COLLECTION Procedure 12-21
CREATE_OR_TRUNCATE_COLLECTION Procedure 12-22
CREATE_COLLECTION_FROM_QUERY Procedure 12-23
CREATE_COLLECTION_FROM_QUERY2 Procedure 12-25
CREATE_COLLECTION_FROM_QUERY_B Procedure 12-27
CREATE_COLLECTION_FROM_QUERYB2 Procedure 12-29
DELETE_ALL_COLLECTIONS Procedure 12-31
DELETE_ALL_COLLECTIONS_SESSION Procedure 12-32
DELETE_COLLECTION Procedure 12-33
DELETE_MEMBER Procedure 12-34
DELETE_MEMBERS Procedure 12-35
GET_MEMBER_MD5 Function 12-36
MERGE_MEMBERS Procedure 12-37
MOVE_MEMBER_DOWN Procedure 12-39

MOVE_MEMBER_UP Procedure 12-40
RESEQUENCE_COLLECTION Procedure 12-41
RESET_COLLECTION_CHANGED Procedure 12-42
RESET_COLLECTION_CHANGED_ALL Procedure 12-43
SORT_MEMBERS Procedure 12-44
TRUNCATE_COLLECTION Procedure 12-45
UPDATE_MEMBER Procedure 12-46
UPDATE_MEMBERS Procedure 12-48
UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1 12-50
UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2 12-52
UPDATE_MEMBER_ATTRIBUTE Procedure Signature 3 12-54
UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4 12-56
UPDATE_MEMBER_ATTRIBUTE Procedure Signature 5 12-58
UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6 12-60

13 APEX_WEB_SERVICE

About the APEX_WEB_SERVICE API 13-2
Invoking a SOAP Style Web Service 13-3
Invoking a RESTful Style Web Service 13-5
Retrieving Cookies and HTTP Headers 13-6
Setting Cookies and HTTP Headers 13-7
BLOB2CLOBBASE64 Function 13-8
CLOBBASE642BLOB Function 13-9
MAKE_REQUEST Procedure 13-10
MAKE_REQUEST Function 13-12
MAKE_REST_REQUEST Function 13-14
PARSE_RESPONSE Function 13-16
PARSE_RESPONSE_CLOB Function 13-17
PARSE_XML Function 13-18
PARSE_XML_CLOB Function 13-20

Index

Preface

Oracle Application Express API Reference describes the Application Programming Interfaces, referred to as APIs, available when programming in the Oracle Application Express environment.

This preface contains these topics:

- Topic Overview
- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Topic Overview

This document contains the following chapters:

Title	Description
APEX_UTIL	Use the `APEX_UTIL` package to get and set session state, get files, check authorizations for users, reset different states for users, and also to get and set preferences for users.
APEX_MAIL	Use the `APEX_MAIL` package to send an email from an Oracle Application Express application.
APEX_ITEM	Use the `APEX_ITEM` package to create form elements dynamically based on a SQL query instead of creating individual items page by page.
APEX_APPLICATION	Use the `APEX_APPLICATION` package to take advantage of a number of global variables.
APEX_CUSTOM_AUTH	Use the `APEX_CUSTOM_AUTH` package to perform various operations related to authentication and session management.
APEX_LDAP	Use `APEX_LDAP` to perform various operations related to Lightweight Directory Access Protocol (LDAP) authentication.
APEX_INSTANCE_ADMIN	Use the `APEX_INSTANCE_ADMIN` package to get and set email settings, wallet settings, report printing settings and to manage schema to workspace mappings.

Title	Description
APEX_UI_DEFAULT_UPDATE	You can use the `APEX_UI_DEFAULT_UPDATE` package to set the user interface defaults associated with a table within a schema. The package must be called from within the schema that owns the table you are updating.
JavaScript APIs	Use these JavaScript functions and objects to provide client-side functionality, such as showing and hiding page elements, or making XML HTTP Asynchronous JavaScript and XML (AJAX) requests.
APEX_PLSQL_JOB	Use `APEX_PLSQL_JOB` package to run PL/SQL code in the background of your application. This is an effective approach for managing long running operations that do not need to complete for a user to continue working with your application.
APEX_LANG	Use `APEX_LANG` API to translate messages.
APEX_COLLECTION	Use `APEX_COLLECTION` to temporarily capture one or more nonscalar values. You can use collections to store rows and columns currently in session state so they can be accessed, manipulated, or processed uring a user's specific session.
APEX_WEB_SERVICE	The APEX_WEB_SERVICE API allows you to integrate other systems with Application Express by allowing you to interact with Web services anywhere you can utilize PL/SQL in your application. The API contains procedures and functions to call both SOAP and RESTful style Web services.

Note: In release 2.2, Oracle Application Express APIs were renamed using the prefix `APEX_`. Note that API's using the previous prefix `HTMLDB_` are still supported to provide backward compatibility. As a best practice, however, use the new API names for new applications unless you plan to run them in an earlier version of Oracle Application Express.

Audience

Oracle Application Express API Reference is intended for application developers who are building database-centric Web applications using Oracle Application Express. The guide describes the APIs available when programming in the Oracle Application Express environment.

To use this guide, you need to have a general understanding of relational database concepts as well as an understanding of the operating system environment under which you are running Oracle Application Express.

See Also: *Oracle 2 Day + Application Express Developer's Guide*

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to

evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at http://www.oracle.com/accessibility/.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/support/contact.html or visit http://www.oracle.com/accessibility/support.html if you are hearing impaired.

Related Documents

For more information, see these Oracle resources:

- *Oracle Application Express Release Notes*
- *Oracle Application Express Installation Guide*
- *Oracle 2 Day + Application Express Developer's Guide*
- *Oracle Application Express Application Builder User's Guide*
- *Oracle Application Express Advanced Tutorials*
- *Oracle Application Express Administration Guide*
- *Oracle Application Express Migration Guide*
- *Oracle Application Express SQL Workshop Guide*
- *Oracle Database Concepts*
- *Oracle Database Advanced Application Developer's Guide*
- *Oracle Database Administrator's Guide*
- *Oracle Database SQL Language Reference*
- *SQL*Plus User's Guide and Reference*
- *Oracle Database PL/SQL Language Reference*

For information about Oracle error messages, see *Oracle Database Error Messages*. Oracle error message documentation is available only in HTML. If you have access to the Oracle Database Documentation Library, you can browse the error messages by range. Once you find the specific range, use your browser's "find in page" feature to locate the specific message. When connected to the Internet, you can search for a specific error message using the error message search feature of the Oracle online documentation.

Many books in the documentation set use the sample schemas of the seed database, which is installed by default when you install Oracle. Refer to *Oracle Database Sample Schemas* for information on how these schemas were created and how you can use them yourself.

For additional documentation available on Oracle's Technology Network, please visit the Oracle Application Express web site located at

`http://www.oracle.com/technology/products/database/application_express/`

For additional application examples, go to the Learning Library. Search for free online training content, including Oracle by Example (OBE), demos, and tutorials. To access the Oracle Learning Library, go to:

`http://apex.oracle.com/pls/apex/f?p=9830:28:0::NO:RIR:P28_PRODUCT_SUITE,IR_P`

Printed documentation is available for sale in the Oracle Store at

`http://shop.oracle.com/`

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

`http://www.oracle.com/technology/membership/`

If you already have a user name and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://www.oracle.com/technology/documentation/`

Conventions

For a description of PL/SQL subprogram conventions, refer to the *Oracle Database PL/SQL Language Reference*. This document contains the following information:

- Specifying subprogram parameter modes
- Specifying default values for subprogram parameters
- Overloading PL/SQL subprogram Names

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New in this Document

This section provides an overview of updates made to this version of the Oracle Application Express API Reference.

Topics:

- Oracle Application Express API Reference Updates for Release 4.0

Oracle Application Express API Reference Updates for Release 4.0

The following APIs have been added to the Oracle Application Express API Reference for Oracle Application Express, release 4.0:

- APEX_COLLECTION

  Collections enable you to temporarily capture one or more nonscalar values. You can use collections to store rows and columns currently in session state so they can be accessed, manipulated, or processed during a user's specific session. You can think of a collection as a bucket in which you temporarily store and name rows of information. See APEX_COLLECTION on page 12-1.

- APEX_WEB_SERVICE

  The APEX_WEB_SERVICE API allows you to integrate other systems with Application Express by allowing you to interact with Web services anywhere you can utilize PL/SQL in your application. The API contains procedures and functions to call both SOAP and RESTful style Web services. It contains functions to parse the responses from Web services and to encode/decode into SOAP friendly base64 encoding. See APEX_WEB_SERVICE on page 13-1

> **See Also:** For an overview of new features in Oracle Application Express, release 4.0, see "What's New" in *Oracle Application Express Application Builder User's Guide*.

1

APEX_UTIL

The APEX_UTIL package provides utilities you can use when programming in the Oracle Application Express environment. You can use the APEX_UTIL package to get and set session state, get files, check authorizations for users, reset different states for users, get and purge cache information and also to get and set preferences for users.

Topics:

- CACHE_GET_DATE_OF_PAGE_CACHE Function
- CACHE_GET_DATE_OF_REGION_CACHE Function
- CACHE_PURGE_BY_APPLICATION Procedure
- CACHE_PURGE_BY_PAGE Procedure
- CACHE_PURGE_STALE Procedure
- CHANGE_CURRENT_USER_PW Procedure
- CHANGE_PASSWORD_ON_FIRST_USE Function
- CLEAR_APP_CACHE Procedure
- CLEAR_PAGE_CACHE Procedure
- CLEAR_USER_CACHE Procedure
- COUNT_CLICK Procedure
- CREATE_USER Procedure
- CREATE_USER_GROUP Procedure
- CURRENT_USER_IN_GROUP Function
- DOWNLOAD_PRINT_DOCUMENT Procedure Signature 1
- DOWNLOAD_PRINT_DOCUMENT Procedure Signature 2
- DOWNLOAD_PRINT_DOCUMENT Procedure Signature 3
- DOWNLOAD_PRINT_DOCUMENT Procedure Signature 4
- EDIT_USER Procedure
- END_USER_ACCOUNT_DAYS_LEFT Function
- EXPIRE_END_USER_ACCOUNT Procedure
- EXPIRE_WORKSPACE_ACCOUNT Procedure
- EXPORT_USERS Procedure
- FETCH_APP_ITEM Function

- FETCH_USER Procedure Signature 1
- FETCH_USER Procedure Signature 2
- FETCH_USER Procedure Signature 3
- FIND_SECURITY_GROUP_ID Function
- FIND_WORKSPACE Function
- GET_ACCOUNT_LOCKED_STATUS Function
- GET_ATTRIBUTE Function
- GET_AUTHENTICATION_RESULT Function
- GET_BLOB_FILE_SRC Function
- GET_CURRENT_USER_ID Function
- GET_DEFAULT_SCHEMA Function
- GET_EDITION Function
- GET_EMAIL Function
- GET_FEEDBACK_FOLLOW_UP Function
- GET_FILE Procedure
- GET_FILE_ID Function
- GET_FIRST_NAME Function
- GET_GROUPS_USER_BELONGS_TO Function
- GET_GROUP_ID Function
- GET_GROUP_NAME Function
- GET_LAST_NAME Function
- GET_NUMERIC_SESSION_STATE Function
- GET_PREFERENCE Function
- GET_PRINT_DOCUMENT Function Signature 1
- GET_PRINT_DOCUMENT Function Signature 2
- GET_PRINT_DOCUMENT Function Signature 3
- GET_PRINT_DOCUMENT Function Signature 4
- GET_SCREEN_READER_MODE_TOGGLE Function
- GET_SESSION_LANG Function
- GET_SESSION_STATE Function
- GET_SESSION_TERRITORY Function
- GET_SESSION_TIME_ZONE Function
- GET_USER_ID Function
- GET_USER_ROLES Function
- GET_USERNAME Function
- IR_CLEAR Procedure
- IR_FILTER Procedure

- IR_RESET Procedure
- IS_LOGIN_PASSWORD_VALID Function
- IS_SCREEN_READER_SESSION Function
- IS_SCREEN_READER_SESSION_YN Function
- IS_USERNAME_UNIQUE Function
- KEYVAL_NUM Function
- KEYVAL_VC2 Function
- LOCK_ACCOUNT Procedure
- PASSWORD_FIRST_USE_OCCURRED Function
- PREPARE_URL Function
- PUBLIC_CHECK_AUTHORIZATION Function
- PURGE_REGIONS_BY_APP Procedure
- PURGE_REGIONS_BY_NAME Procedure
- PURGE_REGIONS_BY_PAGE Procedure
- REMOVE_PREFERENCE Procedure
- REMOVE_SORT_PREFERENCES Procedure
- REMOVE_USER Procedure
- RESET_AUTHORIZATIONS Procedure
- RESET_PW Procedure
- SAVEKEY_NUM Function
- SAVEKEY_VC2 Function
- SET_ATTRIBUTE Procedure
- SET_AUTHENTICATION_RESULT Procedure
- SET_CUSTOM_AUTH_STATUS Procedure
- SET_EDITION Procedure
- SET_EMAIL Procedure
- SET_FIRST_NAME Procedure
- SET_LAST_NAME Procedure
- SET_PREFERENCE Procedure
- SET_SECURITY_GROUP_ID Procedure
- SET_SESSION_LANG Procedure
- SET_SESSION_LIFETIME_SECONDS Procedure
- SET_SESSION_MAX_IDLE_SECONDS Procedure
- SET_SESSION_SCREEN_READER_OFF Procedure
- SET_SESSION_SCREEN_READER_ON Procedure
- SET_SESSION_STATE Procedure
- SET_SESSION_TERRITORY Procedure

- SET_SESSION_TIME_ZONE Procedure
- SET_USERNAME Procedure
- SHOW_SCREEN_READER_MODE_TOGGLE Procedure
- STRING_TO_TABLE Function
- SHOW_SCREEN_READER_MODE_TOGGLE Procedure
- STRONG_PASSWORD_VALIDATION Function
- SUBMIT_FEEDBACK Procedure
- SUBMIT_FEEDBACK_FOLLOWUP Procedure
- TABLE_TO_STRING Function
- UNEXPIRE_END_USER_ACCOUNT Procedure
- UNEXPIRE_WORKSPACE_ACCOUNT Procedure
- UNLOCK_ACCOUNT Procedure
- URL_ENCODE Function
- WORKSPACE_ACCOUNT_DAYS_LEFT Function

CACHE_GET_DATE_OF_PAGE_CACHE Function

This function returns the date and time a specified application page was cached either for the user issuing the call, or for all users if the page was not set to be cached by user.

Syntax

```
APEX_UTIL.CACHE_GET_DATE_OF_PAGE_CACHE (
    p_application  IN    NUMBER,
    p_page         IN    NUMBER)
RETURN DATE;
```

Parameters

Table 1–1 describes the parameters available in the CACHE_GET_DATE_OF_PAGE_CACHE procedure.

Table 1–1 CACHE_GET_DATE_OF_PAGE_CACHE Parameters

Parameter	Description
p_application	The identification number (ID) of the application.
p_page	The page number (ID).

Example

The following example demonstrates how to use the CACHE_GET_DATE_OF_PAGE_CACHE function to retrieve the cache date and time for page 9 of the currently executing application. If page 9 has been cached, the cache date and time is output using the HTP package. The page could have been cached either by the user issuing the call, or for all users if the page was not to be cached by the user.

```
DECLARE
    l_cache_date DATE DEFAULT NULL;
BEGIN
    l_cache_date := APEX_UTIL.CACHE_GET_DATE_OF_PAGE_CACHE(
                        p_application => :APP_ID,
                        p_page => 9);
    IF l_cache_date IS NOT NULL THEN
        HTP.P('Cached on ' || TO_CHAR(l_cache_date, 'DD-MON-YY HH24:MI:SS'));
    END IF;
END;
```

CACHE_GET_DATE_OF_REGION_CACHE Function

This function returns the date and time a specified region was cached either for the user issuing the call, or for all users if the page was not set to be cached by user.

Syntax

```
APEX_UTIL.CACHE_GET_DATE_OF_REGION_CACHE (
    p_application  IN    NUMBER,
    p_page         IN    NUMBER,
    p_region_name  IN    VARCHAR2)
RETURN DATE;
```

Parameters

Table 1–2 describes the parameters available in the CACHE_GET_DATE_OF_REGION_CACHE function.

Table 1–2 CACHE_GET_DATE_OF_REGION_CACHE Parameters

Parameter	Description
p_application	The identification number (ID) of the application
p_page	The page number (ID)
p_region_name	The region name

Example

The following example demonstrates how to use the CACHE_GET_DATE_OF_REGION_CACHE function to retrieve the cache date and time for the region named Cached Region on page 13 of the currently executing application. If the region has been cached, the cache date and time is output using the HTP package. The region could have been cached either by the user issuing the call, or for all users if the page was not to be cached by user.

```
DECLARE
    l_cache_date DATE DEFAULT NULL;
BEGIN
    l_cache_date := APEX_UTIL.CACHE_GET_DATE_OF_REGION_CACHE(
        p_application => :APP_ID,
        p_page => 13,
        p_region_name => 'Cached Region');
    IF l_cache_date IS NOT NULL THEN
        HTP.P('Cached on ' || TO_CHAR(l_cache_date, 'DD-MON-YY HH24:MI:SS'));
    END IF;
END;
```

CACHE_PURGE_BY_APPLICATION Procedure

This procedure purges all cached pages and regions for a given application.

Syntax

```
APEX_UTIL.CACHE_PURGE_BY_APPLICATION (
    p_application  IN  NUMBER);
```

Parameters

Table 1–3 describes the parameters available in the CACHE_PURGE_BY_APPLICATION procedure.

Table 1–3 CACHE_PURGE_BY_APPLICATION Parameters

Parameter	Description
p_application	The identification number (ID) of the application.

Example

The following example demonstrates how to use the CACHE_PURGE_BY_APPLICATION procedure to purge all the cached pages and regions for the application currently executing.

```
BEGIN
    APEX_UTIL.CACHE_PURGE_BY_APPLICATION(p_application => :APP_ID);
END;
```

CACHE_PURGE_BY_PAGE Procedure

This procedure purges the cache for a given application and page. If the page itself is not cached but contains one or more cached regions, then the cache for these will also be purged.

Syntax

```
APEX_UTIL.CACHE_PURGE_BY_PAGE (
    p_application  IN    NUMBER,
    p_page         IN    NUMBER,
    p_user_name    IN    VARCHAR2 DEFAULT NULL);
```

Parameters

Table 1–4 describes the parameters available in the CACHE_PURGE_BY_PAGE procedure.

Table 1–4 CACHE_PURGE_BY_PAGE Parameters

Parameter	Description
p_application	The identification number (ID) of the application.
p_page	The page number (ID).
p_user_name	The user associated with cached pages and regions.

Example

The following example demonstrates how to use the CACHE_PURGE_BY_PAGE procedure to purge the cache for page 9 of the application currently executing. Additionally, if the p_user_name parameter is supplied, this procedure would be further restricted by a specific users cache (only relevant if the cache is set to be by user).

```
BEGIN
    APEX_UTIL.CACHE_PURGE_BY_PAGE(
        p_application => :APP_ID,
        p_page => 9);
END;
```

CACHE_PURGE_STALE Procedure

This procedure deletes all cached pages and regions for a specified application that have passed the defined active time period. When you cache a page or region, you specify an active time period (or Cache Timeout). Once that period has passed, the cache will no longer be used, thus removing those unusable pages or regions from the cache.

Syntax

```
APEX_UTIL.CACHE_PURGE_STALE (
    p_application  IN    NUMBER);
```

Parameters

Table 1–5 describes the parameters available in the CACHE_PURGE_STALE procedure.

Table 1–5 CACHE_PURGE_STALE Parameters

Parameter	Description
p_application	The identification number (ID) of the application.

Example

The following example demonstrates how to use the CACHE_PURGE_STALE procedure to purge all the stale pages and regions in the application currently executing.

```
BEGIN
    APEX_UTIL.CACHE_PURGE_STALE(p_application => :APP_ID);
END;
```

CHANGE_CURRENT_USER_PW Procedure

This procedure changes the password of the currently authenticated user, assuming Application Express user accounts are in use.

Syntax

```
APEX_UTIL.CHANGE_CURRENT_USER_PW(
    p_new_password IN VARCHAR2);
```

Parameters

Table 1–6 describes the parameters available in the CHANGE_CURRENT_USER_PW procedure.

Table 1–6 CHANGE_CURRENT_USER_PW Parameters

Parameter	Description
p_new_password	The new password value in clear text

Example

The following example demonstrates how to use the CHANGE_CURRENT_USER_PW procedure to change the password for the user who is currently authenticated, assuming Application Express accounts are in use.

```
BEGIN
    APEX_UTIL.CHANGE_CURRENT_USER_PW ('secret99');
END;
```

See Also: "RESET_PW Procedure" on page 1-100

CHANGE_PASSWORD_ON_FIRST_USE Function

Enables a developer to check whether this property is enabled or disabled for an end user account. This function returns true if the account password must be changed upon first use (after successful authentication) after the password is initially set and after it is changed on the Administration Service, Edit User page. This function returns false if the account does not have this property.

This function may be run in a page request context by any authenticated user.

Syntax

```
APEX_UTIL.CHANGE_PASSWORD_ON_FIRST_USE (
    p_user_name IN VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 1–7 describes the parameters available in the CHANGE_PASSWORD_ON_FIRST_USE function.

Table 1–7 CHANGE_PASSWORD_ON_FIRST_USE Parameters

Parameter	Description
p_user_name	The user name of the user account

Example

The following example demonstrates how to use the CHANGE_PASSWORD_ON_FIRST_USE function. Use this function to check if the password of an Application Express user account (workspace administrator, developer, or end user) in the current workspace must be changed by the user the first time it is used.

```
BEGIN
    FOR c1 IN (SELECT user_name FROM wwv_flow_users) LOOP
        IF APEX_UTIL.CHANGE_PASSWORD_ON_FIRST_USE(p_user_name => c1.user_name)
THEN
            htp.p('User:'||c1.user_name||' requires password to be changed the
first time it is used.');
        END IF;
    END LOOP;
END;
```

See Also: "PASSWORD_FIRST_USE_OCCURRED Function" on page 1-89

CLEAR_APP_CACHE Procedure

This procedure removes session state for a given application for the current session.

Syntax

```
APEX_UTIL.CLEAR_APP_CACHE (
    p_app_id    IN    VARCHAR2 DEFAULT NULL);
```

Parameters

Table 1–8 describes the parameters available in the CLEAR_APP_CACHE procedure.

Table 1–8 CLEAR_APP_CACHE Parameters

Parameter	Description
p_app_id	The ID of the application for which session state will be cleared for current session

Example

The following example demonstrates how to use the CLEAR_APP_CACHE procedure to clear all the current sessions state for the application with an ID of 100.

```
BEGIN
    APEX_UTIL.CLEAR_APP_CACHE('100');
END;
```

CLEAR_PAGE_CACHE Procedure

This procedure removes session state for a given page for the current session.

Syntax

```
APEX_UTIL.CLEAR_PAGE_CACHE (
    p_page IN NUMBER DEFAULT NULL);
```

Parameters

Table 1–9 describes the parameters available in the CLEAR_PAGE_CACHE procedure.

Table 1–9 CLEAR_PAGE_CACHE Parameters

Parameter	Description
p_page	The ID of the page in the current application for which session state will be cleared for current session.

Example

The following example demonstrates how to use the CLEAR_PAGE_CACHE procedure to clear the current session s state for the page with an ID of 10.

```
BEGIN
    APEX_UTIL.CLEAR_PAGE_CACHE('10');
END;
```

CLEAR_USER_CACHE Procedure

This procedure removes session state and application system preferences for the current user's session. Run this procedure if you reuse session IDs and want to run applications without the benefit of existing session state.

Syntax

```
APEX_UTIL.CLEAR_USER_CACHE;
```

Parameters

None.

Example

The following example demonstrates how to use the CLEAR_USER_CACHE procedure to clear all session state and application system preferences for the current user's session.

```
BEGIN
    APEX_UTIL.CLEAR_USER_CACHE;
END;
```

COUNT_CLICK Procedure

This procedure counts clicks from an application built in Application Builder to an external site. You can also use the shorthand version, procedure `Z`, in place of `APEX_UTIL.COUNT_CLICK`.

Syntax

```
APEX_UTIL.COUNT_CLICK (
    p_url          IN    VARCHAR2,
    p_cat          IN    VARCHAR2,
    p_id           IN    VARCHAR2    DEFAULT NULL,
    p_user         IN    VARCHAR2    DEFAULT NULL,
    p_workspace    IN    VARCHAR2    DEFAULT NULL);
```

Parameters

Table 1–10 describes the parameters available in the `COUNT_CLICK` procedure.

Table 1–10 COUNT_CLICK Parameters

Parameter	Description
`p_url`	The URL to which to redirect
`p_cat`	A category to classify the click
`p_id`	Secondary ID to associate with the click (optional)
`p_user`	The application user ID (optional)
`p_workspace`	The workspace associated with the application (optional)

Example

The following example demonstrates how to use the COUNT_CLICK procedure to log how many user's click on the `http://yahoo.com` link specified. Note that once this information is logged, you can view it via the APEX_WORKSPACE_CLICKS view and in the reports on this view available to workspace and site administrators.

```
DECLARE
    l_url VARCHAR2(255);
    l_cat VARCHAR2(30);
    l_workspace_id VARCHAR2(30);
BEGIN
    l_url := 'http://yahoo.com';
    l_cat := 'yahoo';
    l_workspace_id := TO_CHAR(APEX_UTIL.FIND_SECURITY_GROUP_ID('MY_WORKSPACE'));

    HTP.P('<a href=APEX_UTIL.COUNT_CLICK?p_url=' || l_url || '&p_cat=' || l_cat ||
'&p_workspace=' || l_workspace_id || '>Click</a>');
END;
```

> **See Also:** "FIND_SECURITY_GROUP_ID Function" on page 1-44 in this document and "Purging the External Click Count Log" in *Oracle Application Express Administration Guide*, Defining Authorized URLs in *Oracle Application Express Administration Guide*

CREATE_USER Procedure

This procedure creates a new account record in the Application Express user account table. To execute this procedure, the current user must have administrative privileges.

Syntax

```
APEX_UTIL.CREATE_USER(
    p_user_id                      IN      NUMBER      DEFAULT NULL,
    p_user_name                    IN      VARCHAR2,
    p_first_name                   IN      VARCHAR2    DEFAULT NULL,
    p_last_name                    IN      VARCHAR2    DEFAULT NULL,
    p_description                  IN      VARCHAR2    DEFAULT NULL,
    p_email_address                IN      VARCHAR2    DEFAULT NULL,
    p_web_password                 IN      VARCHAR2,
    p_web_password_format          IN      VARCHAR2    DEFAULT 'CLEAR_TEXT',
    p_group_ids                    IN      VARCHAR2    DEFAULT NULL,
    p_developer_privs              IN      VARCHAR2    DEFAULT NULL,
    p_default_schema               IN      VARCHAR2    DEFAULT NULL,
    p_allow_access_to_schemas      IN      VARCHAR2    DEFAULT NULL,
    p_account_expiry               IN      DATE        DEFAULT TRUNC(SYSDATE),
    p_account_locked               IN      VARCHAR2    DEFAULT 'N',
    p_failed_access_attempts       IN      NUMBER      DEFAULT 0,
    p_change_password_on_first_use IN      VARCHAR2    DEFAULT 'Y',
    p_first_password_use_occurred  IN      VARCHAR2    DEFAULT 'N',
    p_attribute_01                 IN      VARCHAR2    DEFAULT NULL,
    p_attribute_02                 IN      VARCHAR2    DEFAULT NULL,
    p_attribute_03                 IN      VARCHAR2    DEFAULT NULL,
    p_attribute_04                 IN      VARCHAR2    DEFAULT NULL,
    p_attribute_05                 IN      VARCHAR2    DEFAULT NULL,
    p_attribute_06                 IN      VARCHAR2    DEFAULT NULL,
    p_attribute_07                 IN      VARCHAR2    DEFAULT NULL,
    p_attribute_08                 IN      VARCHAR2    DEFAULT NULL,
    p_attribute_09                 IN      VARCHAR2    DEFAULT NULL,
    p_attribute_10                 IN      VARCHAR2    DEFAULT NULL);
```

Parameters

Table 1–11 describes the parameters available in the CREATE_USER procedure.

Table 1–11 CREATE_USER Procedure Parameters

Parameter	Description
p_user_id	Numeric primary key of user account
p_user_name	Alphanumeric name used for login
p_first_name	Informational
p_last_name	Informational
p_description	Informational
p_email_address	Email address
p_web_password	Clear text password
p_web_password_format	If the value your passing for the p_web_password parameter is in clear text format then use CLEAR_TEXT, otherwise use HEX_ENCODED_DIGEST_V2.
p_group_ids	Colon separated list of numeric group IDs

Table 1–11 (Cont.) CREATE_USER Procedure Parameters

Parameter	Description
p_developer_privs	Colon separated list of developer privileges. If p_developer_privs is not null, the user is given access to Team Development. If p_developer_privs contains ADMIN, the user is given Application Builder and SQL Workshop access. If p_developer_privs does not contain ADMIN but contains EDIT, the user is given Application Builder Access. If p_developer_privs does not contain ADMIN but contains SQL, the user is given SQL Workshop access. The following are acceptable values for this parameter: **null** - To create an end user (a user who can only authenticate to developed applications). **CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL** - To create a user with developer privileges with access to Application Builder and SQL Workshop. **ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL** - To create a user with full workspace administrator and developer privileges with access to Application Builder, SQL Workshop and Team Development. Note: Currently this parameter is named inconsistently between the CREATE_USER, EDIT_USER and FETCH_USER APIs, although they all relate to the DEVELOPER_ROLE field stored in the named user account record. CREATE_USER uses p_developer_privs, EDIT_USER uses p_developer_roles and FETCH_USER uses p_developer_role.
p_default_schema	A database schema assigned to the user's workspace, used by default for browsing.
p_allow_access_to_schemas	Colon separated list of schemas assigned to the user's workspace to which the user is restricted (leave null for all).
p_account_expiry	Date password was last updated, which will default to today's date on creation.
p_account_locked	'Y' or 'N' indicating if account is locked or unlocked.
p_failed_access_attempts	Number of consecutive login failures that have occurred, defaults to 0 on creation.
p_change_password_on_first_use	'Y' or 'N' to indicate whether password must be changed on first use, defaults to 'Y' on creation.
p_first_password_use_occurred	'Y' or 'N' to indicate whether login has occurred since password change, defaults to 'N' on creation.
p_attribute_01	Arbitrary text accessible with an API
...	
p_attribute_10	

Example 1

The following simple example creates an 'End User' called 'NEWUSER1' with a password of 'secret99'. Note an 'End User' can only authenticate to developed applications.

```
BEGIN
    APEX_UTIL.CREATE_USER(
        p_user_name    => 'NEWUSER1',
        p_web_password => 'secret99');
```

```
END;
```

Example 2

The following example creates a 'Workspace Administrator' called 'NEWUSER2'. Where the user 'NEWUSER2':

- Has full workspace administration and developer privilege (`p_developer_privs` parameter set to '`ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL`').
- Has access to 2 schemas, both their browsing default '`MY_SCHEMA`' (`p_default_schema` parameter set to '`MY_SCHEMA`') and also '`MY_SCHEMA2`' (`p_allow_access_to_schemas` parameter set to '`MY_SCHEMA2`').
- Does not have to change their password when they first login (`p_change_password_on_first_use` parameter set to 'N').
- Has their phone number stored in the first additional attribute (`p_attribute_01` parameter set to '123 456 7890').

```
BEGIN
    APEX_UTIL.CREATE_USER(
        p_user_name                    => 'NEWUSER2',
        p_first_name                   => 'FRANK',
        p_last_name                    => 'SMITH',
        p_description                  => 'Description...',
        p_email_address                => 'frank@smith.com',
        p_web_password                 => 'password',
        p_developer_privs              => 'ADMIN:CREATE:DATA_
LOADER:EDIT:HELP:MONITOR:SQL',
        p_default_schema               => 'MY_SCHEMA',
        p_allow_access_to_schemas      => 'MY_SCHEMA2',
        p_change_password_on_first_use => 'N',
        p_attribute_01                 => '123 456 7890');
END;
```

See Also: "FETCH_USER Procedure Signature 3" on page 1-41, "EDIT_USER Procedure" on page 1-27, and "GET_GROUP_ID Function" on page 1-61

CREATE_USER_GROUP Procedure

Assuming you are using Application Express authentication, this procedure creates a user group. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
APEX_UTIL.CREATE_USER_GROUP(
    p_id                    IN                  NUMBER,
    p_group_name            IN                  VARCHAR2,
    p_security_group_id     IN                  NUMBER,
    p_group_desc            IN                  VARCHAR2);
```

Parameter

Table 1–12 describes the parameters available in the CREATE_USER_GROUP procedure.

Table 1–12 CREATE_USER_GROUP Parameters

Parameter	Description
p_id	Primary key of group
p_group_name	Name of group
p_security_group_id	Workspace ID
p_group_desc	Descriptive text

Example

The following example demonstrates how to use the CREATE_USER_GROUP procedure to create a new group called 'Managers' with a description of 'text'. Pass null for the p_id parameter to allow the database trigger to assign the new primary key value. Pass null for the p_security_group_id parameter to default to the current workspace ID.

```
BEGIN
    APEX_UTIL.CREATE_USER_GROUP (
        p_id                => null,         -- trigger will assign PK
        p_group_name        => 'Managers',
        p_security_group_id => null,         -- defaults to current workspace ID
        p_group_desc        => 'text');
END;
```

CURRENT_USER_IN_GROUP Function

This function returns a Boolean result based on whether or not the current user is a member of the specified group. You can use the group name or group ID to identify the group.

Syntax

```
APEX_UTIL.CURRENT_USER_IN_GROUP(
    p_group_name    IN VARCHAR2)
RETURN BOOLEAN;

APEX_UTIL.CURRENT_USER_IN_GROUP(
    p_group_id    IN NUMBER)
RETURN BOOLEAN;
```

Parameters

Table 1–13 describes the parameters available in the `CURRENT_USER_IN_GROUP` function.

Table 1–13 CURRENT_USER_IN_GROUP Parameters

Parameter	Description
`p_group_name`	Identifies the name of an existing group in the workspace
`p_group_id`	Identifies the numeric ID of an existing group in the workspace

Example

The following example demonstrates how to use the `CURRENT_USER_IN_GROUP` function to check if the user currently authenticated belongs to the group 'Managers'.

```
DECLARE
    VAL BOOLEAN;
BEGIN
    VAL := APEX_UTIL.CURRENT_USER_IN_GROUP(p_group_name=>'Managers');
END;
```

DOWNLOAD_PRINT_DOCUMENT Procedure Signature 1

This procedure initiates the download of a print document using XML based report data (as a BLOB) and RTF or XSL-FO based report layout.

Syntax

```
APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (
    p_file_name           IN VARCHAR,
    p_content_disposition IN VARCHAR,
    p_report_data         IN BLOB,
    p_report_layout       IN CLOB,
    p_report_layout_type  IN VARCHAR2 default 'xsl-fo',
    p_document_format     IN VARCHAR2 default 'pdf',
    p_print_server        IN VARCHAR2 default null);
```

Parameters

Table 1–14 describes the parameters available in the `DOWNLOAD_PRINT_DOCUMENT` procedure for Signature 1.

Table 1–14 DOWNLOAD_PRINT_DOCUMENT Parameters

Parameter	Description
`p_file_name`	Defines the filename of the print document
`p_content_disposition`	Specifies whether to download the print document or display inline ("attachment", "inline")
`p_report_data`	XML based report data
`p_report_layout`	Report layout in XSL-FO or RTF format
`p_report_layout_type`	Defines the report layout type, that is "xsl-fo" or "rtf"
`p_document_format`	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml"
`p_print_server`	URL of the print server. If not specified, the print server will be derived from preferences.

See Also: "Printing Report Regions" in *Oracle Application Express Application Builder User's Guide*.

DOWNLOAD_PRINT_DOCUMENT Procedure Signature 2

This procedure initiates the download of a print document using pre-defined report query and RTF and XSL-FO based report layout.

Syntax

```
APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (
    p_file_name           IN VARCHAR,
    p_content_disposition IN VARCHAR,
    p_application_id      IN NUMBER,
    p_report_query_name   IN VARCHAR2,
    p_report_layout       IN CLOB,
    p_report_layout_type  IN VARCHAR2 default 'xsl-fo',
    p_document_format     IN VARCHAR2 default 'pdf',
    p_print_server        IN VARCHAR2 default null);
```

Parameters

Table 1–15 describes the parameters available in the DOWNLOAD_PRINT_DOCUMENT function.

Table 1–15 DOWNLOAD_PRINT_DOCUMENT Parameters

Parameter	Description
p_file_name	Defines the filename of the print document
p_content_disposition	Specifies whether to download the print document or display inline ("attachment", "inline")
p_application_id	Defines the application ID of the report query
p_report_query_name	Name of the report query (stored under application's Shared Components)
p_report_layout	Report layout in XSL-FO or RTF format
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf"
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml"
p_print_server	URL of the print server. If not specified, the print server will be derived from preferences.

Example for Signature 2

The following example shows how to use the DOWNLOAD_PRINT_DOCUMENT using Signature 2 (Pre-defined report query and RTF or XSL-FO based report layout.). In this example, the data for the report is taken from a Report Query called 'ReportQueryAndXSL' stored in the current application's Shared Components > Report Queries. The report layout is taken from a value stored in a page item (P1_XSL).

```
BEGIN
    APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (
        p_file_name           => 'mydocument',
        p_content_disposition => 'attachment',
        p_application_id      => :APP_ID,
        p_report_query_name   => 'ReportQueryAndXSL',
        p_report_layout       => :P1_XSL,
```

```
        p_report_layout_type  => 'xsl-fo',
        p_document_format     => 'pdf');
END;
```

See Also: "Printing Report Regions" in *Oracle Application Express Application Builder User's Guide*.

DOWNLOAD_PRINT_DOCUMENT Procedure Signature 3

This procedure initiates the download of a print document using pre-defined report query and pre-defined report layout.

Syntax

```
APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (
    p_file_name           IN VARCHAR,
    p_content_disposition IN VARCHAR,
    p_application_id      IN NUMBER,
    p_report_query_name   IN VARCHAR2,
    p_report_layout_name  IN VARCHAR2,
    p_report_layout_type  IN VARCHAR2 default 'xsl-fo',
    p_document_format     IN VARCHAR2 default 'pdf',
    p_print_server        IN VARCHAR2 default null);
```

Parameters

Table 1–16 describes the parameters available in the DOWNLOAD_PRINT_DOCUMENT procedure for Signature 3.

Table 1–16 DOWNLOAD_PRINT_DOCUMENT Parameters

Parameter	Description
p_file_name	Defines the filename of the print document
p_content_disposition	Specifies whether to download the print document or display inline ("attachment", "inline")
p_application_id	Defines the application ID of the report query
p_report_query_name	Name of the report query (stored under application's Shared Components)
p_report_layout_name	Name of the report layout (stored under application's Shared Components)
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf"
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml"
p_print_server	URL of the print server. If not specified, the print server will be derived from preferences.

Example for Signature 3

The following example shows how to use the DOWNLOAD_PRINT_DOCUMENT using Signature 3 (Pre-defined report query and pre-defined report layout). In this example, the data for the report is taken from a Report Query called 'ReportQuery' stored in the current application's Shared Components > Report Queries. The report layout is taken from a Report Layout called 'ReportLayout' stored in the current application's Shared Components > Report Layouts. Note that if you wish to provision dynamic layouts, instead of specifying 'ReportLayout' for the p_report_layout_name parameter, you could reference a page item that allowed the user to select one of multiple saved Report Layouts. This example also provides a way for the user to specify how they wish to receive the document (as an attachment or inline), through passing the value of P1_CONTENT_DISP to the p_content_disposition

parameter. P1_CONTENT_DISP is a page item of type 'Select List' with the following List of Values Definition:

```
STATIC2:In Browser;inline,Save / Open in separate Window;attachment
```

```
BEGIN
    APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (
        p_file_name           => 'myreport123',
        p_content_disposition => :P1_CONTENT_DISP,
        p_application_id      => :APP_ID,
        p_report_query_name   => 'ReportQuery',
        p_report_layout_name  => 'ReportLayout',
        p_report_layout_type  => 'rtf',
        p_document_format     => 'pdf');
END;
```

See Also: "Printing Report Regions" in *Oracle Application Express Application Builder User's Guide*.

DOWNLOAD_PRINT_DOCUMENT Procedure Signature 4

This procedure initiates the download of a print document using XML based report data (as a CLOB) and RTF or XSL-FO based report layout.

Syntax

```
APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (
    p_file_name           IN VARCHAR,
    p_content_disposition IN VARCHAR,
    p_report_data         IN CLOB,
    p_report_layout       IN CLOB,
    p_report_layout_type  IN VARCHAR2 default 'xsl-fo',
    p_document_format     IN VARCHAR2 default 'pdf',
    p_print_server        IN VARCHAR2 default null);
```

Parameters

Table 1–16 describes the parameters available in the DOWNLOAD_PRINT_DOCUMENT procedure for Signature 4.

Table 1–17 DOWNLOAD_PRINT_DOCUMENT Parameters

Parameter	Description
p_file_name	Defines the filename of the print document
p_content_disposition	Specifies whether to download the print document or display inline ("attachment", "inline")
p_report_data	XML based report data, must be encoded in UTF-8
p_report_layout	Report layout in XSL-FO or RTF format
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf"
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml"
p_print_server	URL of the print server. If not specified, the print server will be derived from preferences.

Example for Signature 4

The following example shows how to use the DOWNLOAD_PRINT_DOCUMENT using Signature 4 (XML based report data (as a CLOB) and RTF or XSL-FO based report layout). In this example both the report data (XML) and report layout (XSL-FO) are taken from values stored in page items.

```
BEGIN
    APEX_UTIL.DOWNLOAD_PRINT_DOCUMENT (
        p_file_name           => 'mydocument',
        p_content_disposition => 'attachment',
        p_report_data         => :P1_XML,
        p_report_layout       => :P1_XSL,
        p_report_layout_type  => 'xsl-fo',
        p_document_format     => 'pdf');
END;
```

See Also: "Printing Report Regions" in *Oracle Application Express Application Builder User's Guide*.

EDIT_USER Procedure

This procedure enables a user account record to be altered. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
APEX_UTIL.EDIT_USER (
    p_user_id                      IN                  NUMBER,
    p_user_name                    IN                  VARCHAR2,
    p_first_name                   IN                  VARCHAR2    DEFAULT NULL,
    p_last_name                    IN                  VARCHAR2    DEFAULT NULL,
    p_web_password                 IN                  VARCHAR2    DEFAULT NULL,
    p_new_password                 IN                  VARCHAR2    DEFAULT NULL,
    p_email_address                IN                  VARCHAR2    DEFAULT NULL,
    p_start_date                   IN                  VARCHAR2    DEFAULT NULL,
    p_end_date                     IN                  VARCHAR2    DEFAULT NULL,
    p_employee_id                  IN                  VARCHAR2    DEFAULT NULL,
    p_allow_access_to_schemas      IN                  VARCHAR2    DEFAULT NULL,
    p_person_type                  IN                  VARCHAR2    DEFAULT NULL,
    p_default_schema               IN                  VARCHAR2    DEFAULT NULL,
    p_group_ids                    IN                  VARCHAR2    DEFAULT NULL,
    p_developer_roles              IN                  VARCHAR2    DEFAULT NULL,
    p_description                  IN                  VARCHAR2    DEFAULT NULL,
    p_account_expiry               IN                  DATE        DEFAULT NULL,
    p_account_locked               IN                  VARCHAR2    DEFAULT 'N',
    p_failed_access_attempts       IN                  NUMBER      DEFAULT 0,
    p_change_password_on_first_use IN                  VARCHAR2    DEFAULT 'Y',
    p_first_password_use_occurred  IN                  VARCHAR2    DEFAULT 'N');
```

Parameters

Table 1–18 describes the parameters available in the EDIT_USER procedure.

Table 1–18 EDIT_USER Parameters

Parameter	Description
p_user_id	Numeric primary key of the user account
p_user_name	Alphanumeric name used for login. **See Also**: "SET_USERNAME Procedure" on page 1-122
p_first_name	Informational. **See Also**: "SET_FIRST_NAME Procedure" on page 1-108
p_last_name	Informational. **See Also**: "SET_LAST_NAME Procedure" on page 1-109
p_web_password	Clear text password. If using this procedure to update the password for the user, values for both p_web_password and p_new_password must not be null and must be identical.
p_new_password	Clear text new password. If using this procedure to update the password for the user, values for both p_web_password and p_new_password must not be null and must be identical.
p_email_address	Informational. **See Also**: "SET_EMAIL Procedure" on page 1-107

Table 1–18 (Cont.) EDIT_USER Parameters

Parameter	Description
p_start_date	Unused
p_end_date	Unused
p_employee_id	Unused
p_allow_access_to_schemas	A list of schemas assigned to the user's workspace to which the user is restricted
p_person_type	Unused
p_default_schema	A database schema assigned to the user's workspace, used by default for browsing
p_group_ids	Colon-separated list of numeric group IDs
p_developer_roles	Colon-separated list of developer privileges. The following are acceptable values for this parameter: ·**null** - To update the user to be an end user (a user who can only authenticate to developed applications) ·**CREATE:DATA_ LOADER:EDIT:HELP:MONITOR:SQL** - To update the user to have developer privilege ·**ADMIN:CREATE:DATA_ LOADER:EDIT:HELP:MONITOR:SQL** - To update the user to have full workspace administrator and developer privilege **Note**: Currently this parameter is named inconsistently between the CREATE_USER, EDIT_USER and FETCH_ USER APIs, although they all relate to the DEVELOPER_ROLE field stored in the named user account record. CREATE_USER uses p_developer_privs, EDIT_USER uses p_developer_roles and FETCH_USER uses p_developer_role. **See Also**: "GET_USER_ROLES Function" on page 1-77
p_description	Informational
p_account_expiry	Date password was last updated. **See Also**: "EXPIRE_END_USER_ACCOUNT Procedure" on page 1-32, "EXPIRE_WORKSPACE_ACCOUNT Procedure" on page 1-33, "UNEXPIRE_END_USER_ ACCOUNT Procedure" on page 1-134, "UNEXPIRE_ WORKSPACE_ACCOUNT Procedure" on page 1-135
p_account_locked	'Y' or 'N' indicating if account is locked or unlocked. **See Also**: "LOCK_ACCOUNT Procedure" on page 1-88, "UNLOCK_ACCOUNT Procedure" on page 1-136
p_failed_access_attempts	Number of consecutive login failures that have occurred.
p_change_password_on_first_ use	'Y' or 'N' to indicate whether password must be changed on first use. **See Also**: "CHANGE_PASSWORD_ON_FIRST_USE Function" on page 1-11
p_first_password_use_ occurred	'Y' or 'N' to indicate whether login has occurred since password change. **See Also**: "PASSWORD_FIRST_USE_OCCURRED Function" on page 1-89

Example

The following example shows how to use the EDIT_USER procedure to update a user account. This example shows how you can use the EDIT_USER procedure to change the user 'FRANK' from a user with just developer privilege to a user with workspace administrator and developer privilege. Firstly, the FETCH_USER procedure is called to assign account details for the user 'FRANK' to local variables. These variables are then used in the call to EDIT_USER to preserve the details of the account, with the exception of the value for the p_developer_roles parameter, which is set to 'ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL'.

```
DECLARE
    l_user_id                       NUMBER;
    l_workspace                     VARCHAR2(255);
    l_user_name                     VARCHAR2(100);
    l_first_name                    VARCHAR2(255);
    l_last_name                     VARCHAR2(255);
    l_web_password                  VARCHAR2(255);
    l_email_address                 VARCHAR2(240);
    l_start_date                    DATE;
    l_end_date                      DATE;
    l_employee_id                   NUMBER(15,0);
    l_allow_access_to_schemas       VARCHAR2(4000);
    l_person_type                   VARCHAR2(1);
    l_default_schema                VARCHAR2(30);
    l_groups                        VARCHAR2(1000);
    l_developer_role                VARCHAR2(60);
    l_description                   VARCHAR2(240);
    l_account_expiry                DATE;
    l_account_locked                VARCHAR2(1);
    l_failed_access_attempts        NUMBER;
    l_change_password_on_first_use  VARCHAR2(1);
    l_first_password_use_occurred   VARCHAR2(1);
BEGIN
    l_user_id := APEX_UTIL.GET_USER_ID('FRANK');

APEX_UTIL.FETCH_USER(
    p_user_id                       => l_user_id,
    p_workspace                     => l_workspace,
    p_user_name                     => l_user_name,
    p_first_name                    => l_first_name,
    p_last_name                     => l_last_name,
    p_web_password                  => l_web_password,
    p_email_address                 => l_email_address,
    p_start_date                    => l_start_date,
    p_end_date                      => l_end_date,
    p_employee_id                   => l_employee_id,
    p_allow_access_to_schemas       => l_allow_access_to_schemas,
    p_person_type                   => l_person_type,
    p_default_schema                => l_default_schema,
    p_groups                        => l_groups,
    p_developer_role                => l_developer_role,
    p_description                   => l_description,
    p_account_expiry                => l_account_expiry,
    p_account_locked                => l_account_locked,
    p_failed_access_attempts        => l_failed_access_attempts,
    p_change_password_on_first_use  => l_change_password_on_first_use,
    p_first_password_use_occurred   => l_first_password_use_occurred);
APEX_UTIL.EDIT_USER (
    p_user_id                       => l_user_id,
    p_user_name                     => l_user_name,
```

```
    p_first_name                    => l_first_name,
    p_last_name                     => l_last_name,
    p_web_password                  => l_web_password,
    p_new_password                  => l_web_password,
    p_email_address                 => l_email_address,
    p_start_date                    => l_start_date,
    p_end_date                      => l_end_date,
    p_employee_id                   => l_employee_id,
    p_allow_access_to_schemas       => l_allow_access_to_schemas,
    p_person_type                   => l_person_type,
    p_default_schema                => l_default_schema,
    p_group_ids                     => l_groups,
    p_developer_roles               => 'ADMIN:CREATE:DATA_
LOADER:EDIT:HELP:MONITOR:SQL',
    p_description                   => l_description,
    p_account_expiry                => l_account_expiry,
    p_account_locked                => l_account_locked,
    p_failed_access_attempts        => l_failed_access_attempts,
    p_change_password_on_first_use  => l_change_password_on_first_use,
    p_first_password_use_occurred   => l_first_password_use_occurred);
END;
```

See Also: "FETCH_USER Procedure Signature 3" on page 1-41

END_USER_ACCOUNT_DAYS_LEFT Function

Returns the number of days remaining before a end user account password expires. This function may be run in a page request context by any authenticated user.

Syntax

```
APEX_UTIL.END_USER_ACCOUNT_DAYS_LEFT (
    p_user_name IN VARCHAR2)
RETURN NUMBER;
```

Parameters

Table 1–19 describes the parameters available in the END_USER_ACCOUNT_DAYS_LEFT function.

Table 1–19 END_USER_ACCOUNT_DAYS_LEFT Parameters

Parameter	Description
p_user_name	The user name of the user account

Example

The following example shows how to use the END_USER_ACCOUNT_DAYS_LEFT function. Use this function to determine the number of days remaining before an Application Express end user account in the current workspace will expire.

```
DECLARE
    l_days_left NUMBER;
BEGIN
    FOR c1 IN (SELECT user_name from wwv_flow_users) LOOP
        l_days_left := APEX_UTIL.END_USER_ACCOUNT_DAYS_LEFT(p_user_name =>
c1.user_name);
        htp.p('End User Account:'||c1.user_name||' will expire in '||l_days_
left||' days.');
    END LOOP;
END;
```

See Also: "EXPIRE_END_USER_ACCOUNT Procedure" on page 1-32 and "UNEXPIRE_END_USER_ACCOUNT Procedure" on page 1-134

EXPIRE_END_USER_ACCOUNT Procedure

Expires the login account for use as a workspace end user. Must be run by an authenticated workspace administrator in a page request context.

Syntax

```
APEX_UTIL.EXPIRE_END_USER_ACCOUNT (
    p_user_name IN VARCHAR2
    );
```

Parameters

Table 1–21 describes the parameters available in the EXPIRE_END_USER_ACCOUNT procedure.

Table 1–20 EXPIRE_END_USER_ACCOUNT Parameters

Parameter	Description
p_user_name	The user name of the user account

Example

The following example shows how to use the EXPIRE_END_USER_ACCOUNT procedure. Use this procedure to expire an Oracle Application Express account (workspace administrator, developer, or end user) in the current workspace. This action specifically expires the account with respect to its use by end users to authenticate to developed applications, but it may also expire the account with respect to its use by developers or administrators to log in to a workspace.

Note that this procedure must be run by a user having administration privileges in the current workspace.

```
BEGIN
    FOR c1 IN (select user_name from wwv_flow_users) LOOP
        APEX_UTIL.EXPIRE_END_USER_ACCOUNT(p_user_name => c1.user_name);
        htp.p('End User Account:'||c1.user_name||' is now expired.');
    END LOOP;
END;
```

See Also: "UNEXPIRE_END_USER_ACCOUNT Procedure" on page 1-134

EXPIRE_WORKSPACE_ACCOUNT Procedure

Expires developer or workspace administrator login accounts. Must be run by an authenticated workspace administrator in a page request context.

Syntax

```
APEX_UTIL.EXPIRE_WORKSPACE_ACCOUNT (
    p_user_name IN VARCHAR2
    );
```

Parameters

Table 1–21 describes the parameters available in the EXPIRE_WORKSPACE_ACCOUNT procedure.

Table 1–21 EXPIRE_WORKSPACE_ACCOUNT Parameters

Parameter	Description
p_user_name	The user name of the user account

Example

The following example shows how to use the EXPIRE_WORKSPACE_ACCOUNT procedure. Use this procedure to expire an Application Express account (workspace administrator, developer, or end user) in the current workspace. This action specifically expires the account with respect to its use by developers or administrators to log in to a workspace, but it may also expire the account with respect to its use by end users to authenticate to developed applications.

```
BEGIN
    FOR c1 IN (SELECT user_name FROM wwv_flow_users) LOOP
        APEX_UTIL.EXPIRE_WORKSPACE_ACCOUNT(p_user_name => c1.user_name);
        htp.p('Workspace Account:'||c1.user_name||' is now expired.');
    END LOOP;
END;
```

See Also: "UNEXPIRE_WORKSPACE_ACCOUNT Procedure" on page 1-135

EXPORT_USERS Procedure

When called from a page, this procedure produces an export file of the current workspace definition, workspace users, and workspace groups. To execute this procedure, the current user must have administrative privilege in the workspace.

Syntax

```
APEX_UTIL.EXPORT_USERS(
    p_export_format IN VARCHAR2 DEFAULT 'UNIX');
```

Parameters

Table 1–22 describes the parameters available in the EXPORT_USERS procedure.

Table 1–22 EXPORT_USERS Parameters

Parameter	Description
p_export_format	Indicates how rows in the export file will be formatted. Specify 'UNIX' to have the resulting file contain rows delimited by line feeds. Specify 'DOS' to have the resulting file contain rows delimited by carriage returns and line feeds

Example

The following example shows how to use the EXPORT_USERS procedure. Call this procedure from a page to produce an export file containing the current workspace definition, list of workspace users and list of workspace groups. The file will be formatted with rows delimited by line feeds.

```
BEGIN
    APEX_UTIL.EXPORT_USERS;
END;
```

FETCH_APP_ITEM Function

This function fetches session state for the current or specified application in the current or specified session.

Syntax

```
APEX_UTIL.FETCH_APP_ITEM(
    p_item    IN VARCHAR2,
    p_app     IN NUMBER DEFAULT NULL,
    p_session IN NUMBER DEFAULT NULL)
RETURN VARCHAR2;
```

Parameters

Table 1–23 describes the parameters available in the `FETCH_APP_ITEM` function.

Table 1–23 FETCH_APP_ITEM Parameters

Parameter	Description
`p_item`	The name of an application-level item (not a page item) whose current value is to be fetched
`p_app`	The ID of the application that owns the item (leave null for the current application)
`p_session`	The session ID from which to obtain the value (leave null for the current session)

Example

The following example shows how to use the `FETCH_APP_ITEM` function to obtain the value of the application item '`F300_NAME`' in application 300. As no value is passed for `p_session`, this defaults to the current session state value.

```
DECLARE
    VAL VARCHAR2(30);
BEGIN
    VAL := APEX_UTIL.FETCH_APP_ITEM(
        p_item => 'F300_NAME',
        p_app => 300);
END;
```

FETCH_USER Procedure Signature 1

This procedure fetches a user account record. To execute this procedure, the current user must have administrative privileges in the workspace. Three overloaded versions of this procedure exist, each with a distinct set of allowed parameters or signatures.

Syntax for Signature 1

```
APEX_UTIL.FETCH_USER (
    p_user_id                      IN                  NUMBER,
    p_workspace                    OUT                 VARCHAR2,
    p_user_name                    OUT                 VARCHAR2,
    p_first_name                   OUT                 VARCHAR2,
    p_last_name                    OUT                 VARCHAR2,
    p_web_password                 OUT                 VARCHAR2,
    p_email_address                OUT                 VARCHAR2,
    p_start_date                   OUT                 VARCHAR2,
    p_end_date                     OUT                 VARCHAR2,
    p_employee_id                  OUT                 VARCHAR2,
    p_allow_access_to_schemas      OUT                 VARCHAR2,
    p_person_type                  OUT                 VARCHAR2,
    p_default_schema               OUT                 VARCHAR2,
    p_groups                       OUT                 VARCHAR2,
    p_developer_role               OUT                 VARCHAR2,
    p_description                  OUT                 VARCHAR2 );
```

Parameters for Signature 1

Table 1–24 describes the parameters available in the FETCH_USER procedure for signature 1.

Table 1–24 Fetch_User Parameters Signature 1

Parameter	Description
p_user_id	Numeric primary key of the user account
p_workspace	The name of the workspace
p_user_name	Alphanumeric name used for login. **See Also**: "GET_USERNAME Function" on page 1-78
p_first_name	Informational. **See Also**: "GET_FIRST_NAME Function" on page 1-59
p_last_name	Informational. **See Also**: "GET_LAST_NAME Function" on page 1-63
p_web_password	Obfuscated account password
p_email_address	Email address. **See Also**: "GET_EMAIL Function" on page 1-54
p_start_date	Unused
p_end_date	Unused
p_employee_id	Unused
p_allow_access_to_schemas	A list of schemas assigned to the user's workspace to which user is restricted
p_person_type	Unused

Table 1–24 (Cont.) Fetch_User Parameters Signature 1

Parameter	Description
p_default_schema	A database schema assigned to the user's workspace, used by default for browsing. **See Also**: "GET_DEFAULT_SCHEMA Function" on page 1-52
p_groups	List of groups of which user is a member. **See Also**: "GET_GROUPS_USER_BELONGS_TO Function" on page 1-60 and "CURRENT_USER_IN_GROUP Function" on page 1-20
p_developer_role	Colon-separated list of developer roles. The following are acceptable values for this parameter: null - Indicates an end user (a user who can only authenticate to developed applications). CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL - Indicates a user with developer privilege. ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL - Indicates a user with full workspace administrator and developer privilege. Note: Currently this parameter is named inconsistently between the CREATE_USER, EDIT_USER and FETCH_USER APIs, although they all relate to the DEVELOPER_ROLE field stored in the named user account record. CREATE_USER uses p_developer_privs, EDIT_USER uses p_developer_roles and FETCH_USER uses p_developer_role. **See Also**: "GET_USER_ROLES Function" on page 1-77
p_description	Informational

Example for Signature 1

The following example shows how to use the FETCH_USER procedure with Signature 1. This procedure is passed the ID of the currently authenticated user for the only IN parameter p_user_id. The code then stores all the other OUT parameter values in local variables.

```
DECLARE
    l_workspace                 VARCHAR2(255);
    l_user_name                 VARCHAR2(100);
    l_first_name                VARCHAR2(255);
    l_last_name                 VARCHAR2(255);
    l_web_password              VARCHAR2(255);
    l_email_address             VARCHAR2(240);
    l_start_date                DATE;
    l_end_date                  DATE;
    l_employee_id               NUMBER(15,0);
    l_allow_access_to_schemas   VARCHAR2(4000);
    l_person_type               VARCHAR2(1);
    l_default_schema            VARCHAR2(30);
    l_groups                    VARCHAR2(1000);
    l_developer_role            VARCHAR2(60);
    l_description               VARCHAR2(240);
BEGIN
    APEX_UTIL.FETCH_USER(
        p_user_id                   => APEX_UTIL.GET_CURRENT_USER_ID,
        p_workspace                 => l_workspace,
        p_user_name                 => l_user_name,
        p_first_name                => l_first_name,
```

```
        p_last_name                => l_last_name,
        p_web_password             => l_web_password,
        p_email_address            => l_email_address,
        p_start_date               => l_start_date,
        p_end_date                 => l_end_date,
        p_employee_id              => l_employee_id,
        p_allow_access_to_schemas  => l_allow_access_to_schemas,
        p_person_type              => l_person_type,
        p_default_schema           => l_default_schema,
        p_groups                   => l_groups,
        p_developer_role           => l_developer_role,
        p_description              => l_description);
END;
```

See Also: "EDIT_USER Procedure" on page 1-27 and "GET_CURRENT_USER_ID Function" on page 1-51

FETCH_USER Procedure Signature 2

This procedure fetches a user account record. To execute this procedure, the current user must have administrative privileges in the workspace. Three overloaded versions of this procedure exist, each with a distinct set of allowed parameters or signatures.

Syntax for Signature 2

```
APEX_UTIL.FETCH_USER (
    p_user_id                      IN                  NUMBER,
    p_user_name                    OUT                 VARCHAR2,
    p_first_name                   OUT                 VARCHAR2,
    p_last_name                    OUT                 VARCHAR2,
    p_email_address                OUT                 VARCHAR2,
    p_groups                       OUT                 VARCHAR2,
    p_developer_role               OUT                 VARCHAR2,
    p_description                  OUT                 VARCHAR2 );
```

Parameters for Signature 2

Table 1–25 describes the parameters available in the FETCH_USER procedure for signature 2.

Table 1–25 Fetch_User Parameters Signature 2

Parameter	Description
p_user_id	Numeric primary key of the user account
p_user_name	Alphanumeric name used for login. **See Also**: "GET_USERNAME Function" on page 1-78
p_first_name	Informational. **See Also:** "GET_FIRST_NAME Function" on page 1-59
p_last_name	Informational. **See Also:** "GET_LAST_NAME Function" on page 1-63
p_email_address	Email address. **See Also**: "GET_EMAIL Function" on page 1-54
p_groups	List of groups of which user is a member. **See Also:** "GET_GROUPS_USER_BELONGS_TO Function" on page 1-60 and "CURRENT_USER_IN_GROUP Function" on page 1-20

Table 1–25 (Cont.) Fetch_User Parameters Signature 2

Parameter	Description
p_developer_role	Colon-separated list of developer roles. The following are acceptable values for this parameter: null - Indicates an end user (a user who can only authenticate to developed applications). CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL - Indicates a user with developer privilege. ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL - Indicates a user with full workspace administrator and developer privilege. Note: Currently this parameter is named inconsistently between the CREATE_USER, EDIT_USER and FETCH_USER APIs, although they all relate to the DEVELOPER_ROLE field stored in the named user account record. CREATE_USER uses p_developer_privs, EDIT_USER uses p_developer_roles and FETCH_USER uses p_developer_role. **See Also**: "GET_USER_ROLES Function" on page 1-77
p_description	Informational

Example for Signature 2

The following example shows how to use the FETCH_USER procedure with Signature 2. This procedure is passed the ID of the currently authenticated user for the only IN parameter p_user_id. The code then stores all the other OUT parameter values in local variables.

```
DECLARE
    l_user_name         VARCHAR2(100);
    l_first_name        VARCHAR2(255);
    l_last_name         VARCHAR2(255);
    l_email_address     VARCHAR2(240);
    l_groups            VARCHAR2(1000);
    l_developer_role    VARCHAR2(60);
    l_description       VARCHAR2(240);
BEGIN
    APEX_UTIL.FETCH_USER(
        p_user_id           => APEX_UTIL.GET_CURRENT_USER_ID,
        p_user_name         => l_user_name,
        p_first_name        => l_first_name,
        p_last_name         => l_last_name,
        p_email_address     => l_email_address,
        p_groups            => l_groups,
        p_developer_role    => l_developer_role,
        p_description       => l_description);
END;
```

See Also: "EDIT_USER Procedure" on page 1-27 and "GET_CURRENT_USER_ID Function" on page 1-51

FETCH_USER Procedure Signature 3

This procedure fetches a user account record. To execute this procedure, the current user must have administrative privileges in the workspace. Three overloaded versions of this procedure exist, each with a distinct set of allowed parameters or signatures.

Syntax for Signature 3

```
APEX_UTIL.FETCH_USER (
    p_user_id                       IN              NUMBER,
    p_workspace                     OUT             VARCHAR2,
    p_user_name                     OUT             VARCHAR2,
    p_first_name                    OUT             VARCHAR2,
    p_last_name                     OUT             VARCHAR2,
    p_web_password                  OUT             VARCHAR2,
    p_email_address                 OUT             VARCHAR2,
    p_start_date                    OUT             VARCHAR2,
    p_end_date                      OUT             VARCHAR2,
    p_employee_id                   OUT             VARCHAR2,
    p_allow_access_to_schemas       OUT             VARCHAR2,
    p_person_type                   OUT             VARCHAR2,
    p_default_schema                OUT             VARCHAR2,
    p_groups                        OUT             VARCHAR2,
    p_developer_role                OUT             VARCHAR2,
    p_description                   OUT             VARCHAR2,
    p_account_expiry                OUT             DATE,
    p_account_locked                OUT             VARCHAR2,
    p_failed_access_attempts        OUT             NUMBER,
    p_change_password_on_first_use  OUT             VARCHAR2,
    p_first_password_use_occurred   OUT             VARCHAR2 );
```

Parameters for Signature 3

Table 1–26 describes the parameters available in the FETCH_USER procedure.

Table 1–26 Fetch_User Parameters Signature 3

Parameter	Description
p_user_id	Numeric primary key of the user account
p_workspace	The name of the workspace
p_user_name	Alphanumeric name used for login. **See Also**: "GET_USERNAME Function" on page 1-78
p_first_name	Informational. **See Also**: "GET_FIRST_NAME Function" on page 1-59
p_last_name	Informational. **See Also**: "GET_LAST_NAME Function" on page 1-63
p_web_password	Obfuscated account password
p_email_address	Email address. **See Also**: "GET_EMAIL Function" on page 1-54
p_start_date	Unused
p_end_date	Unused

Table 1–26 (Cont.) Fetch_User Parameters Signature 3

Parameter	Description
p_employee_id	Unused
p_allow_access_to_schemas	A list of schemas assigned to the user's workspace to which user is restricted
p_person_type	Unused
p_default_schema	A database schema assigned to the user's workspace, used by default for browsing. **See Also**: "GET_DEFAULT_SCHEMA Function" on page 1-52
p_groups	List of groups of which user is a member. **See Also**: "GET_GROUPS_USER_BELONGS_TO Function" on page 1-60 and "CURRENT_USER_IN_GROUP Function" on page 1-20
p_developer_role	Colon-separated list of developer roles. The following are acceptable values for this parameter: null - Indicates an end user (a user who can only authenticate to developed applications). CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL - Indicates a user with developer privilege. ADMIN:CREATE:DATA_LOADER:EDIT:HELP:MONITOR:SQL - Indicates a user with full workspace administrator and developer privilege. Note: Currently this parameter is named inconsistently between the CREATE_USER, EDIT_USER and FETCH_USER APIs, although they all relate to the DEVELOPER_ROLE field stored in the named user account record. CREATE_USER uses p_developer_privs, EDIT_USER uses p_developer_roles and FETCH_USER uses p_developer_role. **See Also**: "GET_USER_ROLES Function" on page 1-77
p_description	Informational
p_account_expiry	Date account password was last reset. **See Also**: "END_USER_ACCOUNT_DAYS_LEFT Function" on page 1-31 and "WORKSPACE_ACCOUNT_DAYS_LEFT Function" on page 1-139
p_account_locked	Locked/Unlocked indicator Y or N. **See Also**: "GET_ACCOUNT_LOCKED_STATUS Function" on page 1-46
p_failed_access_attempts	Counter for consecutive login failures
p_change_password_on_first_use	Setting to force password change on first use Y or N
p_first_password_use_occurred	Indicates whether login with password occurred Y or N

Example for Signature 3

The following example shows how to use the FETCH_USER procedure with Signature 3. This procedure is passed the ID of the currently authenticated user for the only IN parameter p_user_id. The code then stores all the other OUT parameter values in local variables.

```
DECLARE
    l_workspace                     VARCHAR2(255);
    l_user_name                     VARCHAR2(100);
    l_first_name                    VARCHAR2(255);
    l_last_name                     VARCHAR2(255);
    l_web_password                  VARCHAR2(255);
    l_email_address                 VARCHAR2(240);
    l_start_date                    DATE;
    l_end_date                      DATE;
    l_employee_id                   NUMBER(15,0);
    l_allow_access_to_schemas       VARCHAR2(4000);
    l_person_type                   VARCHAR2(1);
    l_default_schema                VARCHAR2(30);
    l_groups                        VARCHAR2(1000);
    l_developer_role                VARCHAR2(60);
    l_description                   VARCHAR2(240);
    l_account_expiry                DATE;
    l_account_locked                VARCHAR2(1);
    l_failed_access_attempts        NUMBER;
    l_change_password_on_first_use  VARCHAR2(1);
    l_first_password_use_occurred   VARCHAR2(1);
BEGIN
    APEX_UTIL.FETCH_USER(
        p_user_id                       => APEX_UTIL.GET_CURRENT_USER_ID,
        p_workspace                     => l_workspace,
        p_user_name                     => l_user_name,
        p_first_name                    => l_first_name,
        p_last_name                     => l_last_name,
        p_web_password                  => l_web_password,
        p_email_address                 => l_email_address,
        p_start_date                    => l_start_date,
        p_end_date                      => l_end_date,
        p_employee_id                   => l_employee_id,
        p_allow_access_to_schemas       => l_allow_access_to_schemas,
        p_person_type                   => l_person_type,
        p_default_schema                => l_default_schema,
        p_groups                        => l_groups,
        p_developer_role                => l_developer_role,
        p_description                   => l_description,
        p_account_expiry                => l_account_expiry,
        p_account_locked                => l_account_locked,
        p_failed_access_attempts        => l_failed_access_attempts,
        p_change_password_on_first_use  => l_change_password_on_first_use,
        p_first_password_use_occurred   => l_first_password_use_occurred);
END;
```

See Also: "EDIT_USER Procedure" on page 1-27 and "GET_CURRENT_USER_ID Function" on page 1-51

FIND_SECURITY_GROUP_ID Function

This function returns the numeric security group ID of the named workspace.

Syntax

```
APEX_UTIL.FIND_SECURITY_GROUP_ID(
    p_workspace    IN VARCHAR2)
RETURN NUMBER;
```

Parameters

Table 1–27 describes the parameters available in the FIND_SECURITY_GROUP_ID function.

Table 1–27 FIND_SECURITY_GROUP_ID Parameters

Parameter	Description
p_workspace	The name of the workspace

Example

The following example demonstrates how to use the FIND_SECURITY_GROUP_ID function to return the security group ID for the workspace called 'DEMOS'.

```
DECLARE
    VAL NUMBER;
BEGIN
    VAL := APEX_UTIL.FIND_SECURITY_GROUP_ID (p_workspace=>'DEMOS');
END;
```

FIND_WORKSPACE Function

This function returns the workspace name associated with a security group ID.

Syntax

```
APEX_UTIL.FIND_WORKSPACE(
    p_security_group_id    IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 1–28 describes the parameters available in the FIND_WORKSPACE function.

Table 1–28 FIND_WORKSPACE Parameters

Parameter	Description
p_security_group_id	The security group ID of a workspace

Example

The following example demonstrates how to use the FIND_WORKSPACE function to return the workspace name for the workspace with a security group ID of 20.

```
DECLARE
    VAL VARCHAR2(255);
BEGIN
    VAL := APEX_UTIL.FIND_WORKSPACE (p_security_group_id =>'20');
END;
```

GET_ACCOUNT_LOCKED_STATUS Function

Returns TRUE if the account is locked and FALSE if the account is unlocked. Must be run by an authenticated workspace administrator in a page request context.

Syntax

```
APEX_UTIL.GET_ACCOUNT_LOCKED_STATUS (
     p_user_name IN VARCHAR2
     ) RETURN BOOLEAN;
```

Parameters

Table 1–29 describes the parameters available in the GET_ACCOUNT_LOCKED_STATUS function.

Table 1–29 GET_ACCOUNT_LOCKED_STATUS Parameters

Parameter	Description
p_user_name	The user name of the user account

Example

The following example shows how to use the GET_ACCOUNT_LOCKED_STATUS function. Use this function to check if an Application Express user account (workspace administrator, developer, or end user) in the current workspace is locked.

```
BEGIN
    FOR c1 IN (SELECT user_name FROM wwv_flow_users) loop
        IF APEX_UTIL.GET_ACCOUNT_LOCKED_STATUS(p_user_name => c1.user_name) THEN
            HTP.P('User Account:'||c1.user_name||' is locked.');
        END IF;
    END LOOP;
END;
```

See Also: LOCK_ACCOUNT Procedure on page 1-88 and UNLOCK_ACCOUNT Procedure on page 1-136.

GET_ATTRIBUTE Function

This function returns the value of one of the attribute values (1 through 10) of a named user in the Application Express accounts table. Please note these are only accessible via the APIs.

Syntax

```
APEX_UTIL.GET_ATTRIBUTE(
    p_username                IN VARCHAR2,
    p_attribute_number        IN NUMBER)
RETURN VARCHAR2;
```

Parameters

Table 1–30 describes the parameters available in the GET_ATTRIBUTE function.

Table 1–30 GET_ATTRIBUTE Parameters

Parameter	Description
p_username	User name in the account.
p_attribute_number	Number of attributes in the user record (1 through 10)

Example

The following example shows how to use the GET_ATTTIBUTE function to return the value for the 1st attribute for the user 'FRANK'.

```
DECLARE
    VAL VARCHAR2(4000);
BEGIN
    VAL := APEX_UTIL.GET_ATTRIBUTE (
        p_username => 'FRANK',
        p_attribute_number => 1);
END;
```

See Also: "SET_ATTRIBUTE Procedure" on page 1-103

GET_AUTHENTICATION_RESULT Function

Use this function to retrieve the authentication result of the current session. Any authenticated user can call this function in a page request context.

Syntax

```
APEX_UTIL.GET_AUTHENTICATION_RESULT
RETURN NUMBER;
```

Parameters

None.

Example

The following example demonstrates how to use the post-authentication process of an application's authentication scheme to retrieve the authentication result code set during authentication.

```
APEX_UTIL.SET_SESSION_STATE('MY_AUTH_STATUS','Authentication result:'||APEX_
UTIL.GET_AUTHENTICATION_RESULT);
```

See Also: "SET_AUTHENTICATION_RESULT Procedure" on page 1-104 and "SET_CUSTOM_AUTH_STATUS Procedure" on page 1-105

GET_BLOB_FILE_SRC Function

As an alternative to using the built-in methods of providing a download link, you can use the APEX_UTIL.GET_BLOB_FILE_SRC function. One advantage of this approach, is the ability to more specifically format the display of the image (with height and width tags). Please note that this approach is only valid if called from a valid Oracle Application Express session. Also, this method requires that the parameters that describe the BLOB to be listed as the format of a valid item within the application. That item is then referenced by the function.

> **See Also:** "About BLOB Support in Forms and Reports" in *Oracle Application Express Application Builder User's Guide*

Syntax

```
APEX_UTIL.GET_BLOB_FILE_SRC (
    p_item_name           IN VARCHAR2 DEFAULT NULL,
    p_v1                  IN VARCHAR2 DEFAULT NULL,
    p_v2                  IN VARCHAR2 DEFAULT NULL,
    p_content_disposition IN VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;
```

Parameters

Table 1–31 describes the parameters available in GET_BLOB_FILE_SRC function.

Table 1–31 GET_BLOB_FILE_SRC Parameters

Parameter	Description
p_item_name	Name of valid application page ITEM that with type FILE that contains the source type of DB column.
p_v1	Value of primary key column 1.
p_v2	Value of primary key column 2.
p_content_disposition	Specify inline or attachment, all other values ignored

Example

As a PLSQL Function Body:

```
RETURN '<img src="'||APEX_UTIL.GET_BLOB_FILE_SRC('P2_ATTACHMENT',:P2_EMPNO)||'"
/>';
```

As a Region Source of type SQL:

```
SELECT ID, NAME,CASE WHEN NVL(dbms_lob.getlength(document),0) = 0
    THEN NULL
    ELSE CASE WHEN attach_mimetype like 'image%'
    THEN '<img src="'||apex_util.get_blob_file_src('P4_DOCUMENT',id)||'" />'
    ELSE
    '<a href="'||apex_util.get_blob_file_src('P4_DOCUMENT',id)||'">Download</a>'
    end
    END new_img
    FROM TEST_WITH_BLOB
```

The previous example illustrates how to display the BLOB within the report, if it can be displayed, and provide a download link, if it cannot be displayed.

See Also: "Running a Demonstration Application" in *Oracle Application Express Application Builder User's Guide*.

GET_CURRENT_USER_ID Function

This function returns the numeric user ID of the current user.

Syntax

```
APEX_UTIL.GET_CURRENT_USER_ID
RETURN NUMBER;
```

Parameters

None.

Example

This following example shows how to use the `GET_CURRENT_USER_ID` function. It returns the numeric user ID of the current user into a local variable.

```
DECLARE
    VAL NUMBER;
BEGIN
    VAL := APEX_UTIL.GET_CURRENT_USER_ID;
END;
```

GET_DEFAULT_SCHEMA Function

This function returns the default schema name associated with the current user.

Syntax

```
APEX_UTIL.GET_DEFAULT_SCHEMA
RETURN VARCHAR2;
```

Parameters

None.

Example

The following example shows how to use the `GET_DEFAULT_SCHEMA` function. It returns the default schema name associated with the current user into a local variable.

```
DECLARE
    VAL VARCHAR2(30);
BEGIN
    VAL := APEX_UTIL.GET_DEFAULT_SCHEMA;
END;
```

GET_EDITION Function

This function returns the edition for the current page view.

Syntax

```
APEX_UTIL.GET_EDITION
RETURN VARCHAR2;
```

Parameters

None.

Example

The following example shows how to use the `GET_EDITION` function. It returns the edition name for the current page view into a local variable.

```
DECLARE
    VAL VARCHAR2(30);
BEGIN
    VAL := APEX_UTIL.GET_EDITION;
END;
```

GET_EMAIL Function

This function returns the email address associated with the named user.

Syntax

```
APEX_UTIL.GET_EMAIL(
   p_username IN VARCHAR2);
RETURN VARCHAR2;
```

Parameters

Table 1–32 describes the parameters available in GET_EMAIL function.

Table 1–32 GET_EMAIL Parameters

Parameter	Description
p_username	The user name in the account

Example

The following example shows how to use the GET_EMAIL function to return the email address of the user 'FRANK'.

```
DECLARE
    VAL VARCHAR2(240);
BEGIN
    VAL := APEX_UTIL.GET_EMAIL(p_username => 'FRANK');
END;
```

See Also: "SET_EMAIL Procedure" on page 1-107

GET_FEEDBACK_FOLLOW_UP Function

Syntax

```
APEX_UTIL.GET_EMAIL(
   p_username IN VARCHAR2);
RETURN VARCHAR2;

APEX_UTIL.GET_FEEDBACK_FOLLOW_UP (
    p_feedback_id    IN NUMBER,
    p_row            IN NUMBER DEFAULT 1,
    p_template       IN VARCHAR2 DEFAULT '<br />#CREATED_ON# (#CREATED_BY#)
#FOLLOW_UP#')
RETURN VARCHAR2;
```

Parameters

Table 1–33 describes the parameters available in `GET_FEEDBACK_FOLLOW_UP` function.

Table 1–33 GET_FEEDBACK_FOLLOW_UP Parameters

Parameter	Description
`p_feedback_id`	The unique identifier of the feedback item.
`p_row`	Identifies which followup to retrieve and is ordered by created_on_desc.
`p_template`	The template to use to return the follow up. Given the in the default template, the function can be used in a loop to return all the follow up to a feedback.

GET_FILE Procedure

This procedure downloads files from the Oracle Application Express file repository. Please note if you are invoking this procedure during page processing, you must ensure that no page branch will be invoked under the same condition, as it will interfere with the file retrieval. This means that branches with any of the following conditions should not be set to fire:

- Branches with a 'When Button Pressed' attribute equal to the button that invokes the procedure.
- Branches with conditional logic defined that would succeed during page processing when the procedure is being invoked.
- As unconditional.

Syntax

```
APEX_UTIL.GET_FILE (
    p_file_id    IN   VARCHAR2,
    p_inline     IN   VARCHAR2 DEFAULT 'NO');
```

Parameters

Table 1–34 describes the parameters available in GET_FILE procedure.

Table 1–34 GET_FILE Parameters

Parameter	Description

p_file_id	ID in APEX_APPLICATION_FILES of the file to be downloaded. APEX_APPLICATION_FILES is a view on all files uploaded to your workspace. The following example demonstrates how to use APEX_APPLICATION_FILES:
	`DECLARE`
	`  l_file_id NUMBER;`
	`BEGIN`
	`  SELECT id`
	`  INTO l_file_id`
	`  FROM APEX_APPLICATION_FILES`
	`  WHERE filename = 'myxml';`
	`  --`
	`  APEX_UTIL.GET_FILE(`
	`  p_file_id => l_file_id,`
	`  p_inline => 'YES');`
	`END;`

p_inline	Valid values include YES and NO. YES to display inline in a browser. NO to download as attachment

Example

The following example shows how to use the GET_FILE function to return the file identified by the ID 8675309. This will be displayed inline in the browser.

```
BEGIN
    APEX_UTIL.GET_FILE(
        p_file_id   => '8675309',
        p_inline    => 'YES');
```

```
END;
```

See Also: "GET_FILE_ID Function" on page 1-58

GET_FILE_ID Function

This function obtains the primary key of a file in the Oracle Application Express file repository.

Syntax

```
APEX_UTIL.GET_FILE_ID (
    p_name    IN   VARCHAR2)
RETURN NUMBER;
```

Parameters

Table 1–35 describes the parameters available in GET_FILE_ID function.

Table 1–35 GET_FILE_ID Parameters

Parameter	Description
p_name	The NAME in APEX_APPLICATION_FILES of the file to be downloaded. APEX_APPLICATION_FILES is a view on all files uploaded to your workspace.

Example

The following example shows how to use the GET_FILE_ID function to retrieve the database ID of the file with a filename of 'F125.sql'.

```
DECLARE
    l_name VARCHAR2(255);
    l_file_id NUMBER;
BEGIN
    SELECT name
        INTO l_name
        FROM APEX_APPLICATION_FILES
        WHERE filename = 'F125.sql';
        --
        l_file_id := APEX_UTIL.GET_FILE_ID(p_name => l_name);
END;
```

GET_FIRST_NAME Function

This function returns the FIRST_NAME field stored in the named user account record.

Syntax

```
APEX_UTIL.GET_FIRST_NAME
    p_username IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 1–36 describes the parameters available in GET_FIRST_NAME function.

Table 1–36 GET_FIRST_NAME Parameters

Parameter	Description
p_username	Identifies the user name in the account

Example

The following example shows how to use the GET_FIRST_NAME function to return the FIRST_NAME of the user 'FRANK'.

```
DECLARE
    VAL VARCHAR2(255);
BEGIN
    VAL := APEX_UTIL.GET_FIRST_NAME(p_username => 'FRANK');
END;
```

See Also: "SET_FIRST_NAME Procedure" on page 1-108

GET_GROUPS_USER_BELONGS_TO Function

This function returns a comma then a space separated list of group names to which the named user is a member.

Syntax

```
APEX_UTIL.GET_GROUPS_USER_BELONGS_TO(
   p_username IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 1–37 describes the parameters available in GET_GROUPS_USER_BELONGS_TO function.

Table 1–37 GET_GROUPS_USER_BELONGS_TO Parameters

Parameter	Description
p_username	Identifies the user name in the account

Example

The following example shows how to use the GET_GROUPS_USER_BELONGS_TO to return the list of groups to which the user 'FRANK' is a member.

```
DECLARE
    VAL VARCHAR2(32765);
BEGIN
    VAL := APEX_UTIL.GET_GROUPS_USER_BELONGS_TO(p_username => 'FRANK');
END;
```

See Also: "EDIT_USER Procedure" on page 1-27

GET_GROUP_ID Function

This function returns the numeric ID of a named group in the workspace.

Syntax

```
APEX_UTIL.GET_GROUP_ID(
    p_group_name IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 1–38 describes the parameters available in GET_GROUP_ID function.

Table 1–38 GET_GROUP_ID Parameters

Parameter	Description
p_group_name	Identifies the user name in the account

Example

The following example shows how to use the GET_GROUP_ID function to return the ID for the group named 'Managers'.

```
DECLARE
    VAL NUMBER;
BEGIN
    VAL := APEX_UTIL.GET_GROUP_ID(p_group_name => 'Managers');
END;
```

GET_GROUP_NAME Function

This function returns the name of a group identified by a numeric ID.

Syntax

```
APEX_UTIL.GET_GROUP_NAME(
    p_group_id IN NUMBER)
RETURN VARCHAR2;
```

Parameters

Table 1–39 describes the parameters available in GET_GROUP_NAME function.

Table 1–39 GET_GROUP_NAME Parameters

Parameter	Description
p_group_id	Identifies a numeric ID of a group in the workspace

Example

The following example shows how to use the GET_GROUP_NAME function to return the name of the group with the ID 8922003.

```
DECLARE
    VAL VARCHAR2(255);
BEGIN
    VAL := APEX_UTIL.GET_GROUP_NAME(p_group_id => 8922003);
END;
```

GET_LAST_NAME Function

This function returns the LAST_NAME field stored in the named user account record.

Syntax

```
APEX_UTIL.GET_LAST_NAME(
    p_username IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 1–40 describes the parameters available in GET_LAST_NAME function.

Table 1–40 GET_LAST_NAME Parameters

Parameter	Description
p_username	The user name in the user account record

Example

The following example shows how to use the function to return the LAST_NAME for the user 'FRANK'.

```
DECLARE
    VAL VARCHAR2(255);
BEGIN
    VAL := APEX_UTIL.GET_LAST_NAME(p_username => 'FRANK');
END;
```

See Also: "SET_LAST_NAME Procedure" on page 1-109

GET_NUMERIC_SESSION_STATE Function

This function returns a numeric value for a numeric item. You can use this function in Oracle Application Express applications wherever you can use PL/SQL or SQL. You can also use the shorthand, function NV, in place of APEX_UTIL.GET_NUMERIC_SESSION_STATE.

Syntax

```
APEX_UTIL.GET_NUMERIC_SESSION_STATE (
    p_item     IN VARCHAR2)
RETURN NUMBER;
```

Parameters

Table 1–41 describes the parameters available in GET_NUMERIC_SESSION_STATE function.

Table 1–41 GET_NUMERIC_SESSION_STATE Parameters

Parameter	Description
p_item	The case insensitive name of the item for which you want to have the session state fetched

Example

The following example shows how to use the function to return the numeric value stored in session state for the item 'my_item'.

```
DECLARE
    l_item_value    NUMBER;
BEGIN
    l_item_value := APEX_UTIL.GET_NUMERIC_SESSION_STATE('my_item');
END;
```

See Also: "GET_SESSION_STATE Function" on page 1-73 and "SET_SESSION_STATE Procedure" on page 1-119

GET_PREFERENCE Function

This function retrieves the value of a previously saved preference for a given user.

Syntax

```
APEX_UTIL.GET_PREFERENCE (
    p_preference  IN    VARCHAR2 DEFAULT NULL,
    p_user        IN    VARCHAR2 DEFAULT V('USER'))
RETURN VARCHAR2;
```

Parameters

Table 1–42 describes the parameters available in the `GET_PREFERENCE` function.

Table 1–42 GET_PREFERENCE Parameters

Parameter	Description
`p_preference`	Name of the preference to retrieve the value
`p_value`	Value of the preference
`p_user`	User for whom the preference is being retrieved

Example

The following example shows how to use the `GET_PREFERENCE` function to return the value for the currently authenticated user's preference named '`default_view`'.

```
DECLARE
    l_default_view    VARCHAR2(255);
BEGIN
    l_default_view := APEX_UTIL.GET_PREFERENCE(
        p_preference => 'default_view',
        p_user       => :APP_USER);
END;
```

> **See Also:** "SET_PREFERENCE Procedure" on page 1-110, "REMOVE_PREFERENCE Procedure" on page 1-96 and "Managing User Preferences" in *Oracle Application Express Administration Guide*.

GET_PRINT_DOCUMENT Function Signature 1

This function returns a document as BLOB using XML based report data and RTF or XSL-FO based report layout.

Syntax

```
APEX_UTIL.GET_PRINT_DOCUMENT (
    p_report_data          IN BLOB,
    p_report_layout        IN CLOB,
    p_report_layout_type   IN VARCHAR2 default 'xsl-fo',
    p_document_format      IN VARCHAR2 default 'pdf',
    p_print_server         IN VARCHAR2 default NULL)
RETURN BLOB;
```

Parameters

Table 1–43 describes the parameters available in the GET_PRINT_DOCUMENT function.

Table 1–43 GET_PRINT_DOCUMENT Parameters

Parameter	Description
p_report_data	XML based report data
p_report_layout	Report layout in XSL-FO or RTF format
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf"
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml"
p_print_server	URL of the print server. If not specified, the print server will be derived from preferences.

For a GET_PRINT_DOCUMENT example see "GET_PRINT_DOCUMENT Function Signature 4".

GET_PRINT_DOCUMENT Function Signature 2

This function returns a document as BLOB using pre-defined report query and pre-defined report layout.

Syntax

```
APEX_UTIL.GET_PRINT_DOCUMENT (
    p_application_id      IN NUMBER,
    p_report_query_name   IN VARCHAR2,
    p_report_layout_name  IN VARCHAR2 default null,
    p_report_layout_type  IN VARCHAR2 default 'xsl-fo',
    p_document_format     IN VARCHAR2 default 'pdf',
    p_print_server        IN VARCHAR2 default null)
RETURN BLOB;
```

Parameters

Table 1–44 describes the parameters available in the GET_PRINT_DOCUMENT function.

Table 1–44 GET_PRINT_DOCUMENT Parameters

Parameter	Description
p_application_id	Defines the application ID of the report query
p_report_query_name	Name of the report query (stored under application's shared components)
p_report_layout_name	Name of the report layout (stored under application's Shared Components)
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf"
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml"
p_print_server	URL of the print server. If not specified, the print server will be derived from preferences.

For a GET_PRINT_DOCUMENT example see "GET_PRINT_DOCUMENT Function Signature 4".

GET_PRINT_DOCUMENT Function Signature 3

This function returns a document as BLOB using a pre-defined report query and RTF or XSL-FO based report layout.

Syntax

```
APEX_UTIL.GET_PRINT_DOCUMENT (
    p_application_id      IN NUMBER,
    p_report_query_name   IN VARCHAR2,
    p_report_layout       IN CLOB,
    p_report_layout_type  IN VARCHAR2 default 'xsl-fo',
    p_document_format     IN VARCHAR2 default 'pdf',
    p_print_server        IN VARCHAR2 default null)
RETURN BLOB;
```

Parameters

Table 1–45 describes the parameters available in the GET_PRINT_DOCUMENT function.

Table 1–45 GET_PRINT_DOCUMENT Parameters

Parameter	Description
p_application_id	Defines the application ID of the report query
p_report_query_name	Name of the report query (stored under application's shared components)
p_report_layout	Defines the report layout in XSL-FO or RTF format
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf"
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml"
p_print_server	URL of the print server. If not specified, the print server will be derived from preferences.

For a GET_PRINT_DOCUMENT example see "GET_PRINT_DOCUMENT Function Signature 4".

GET_PRINT_DOCUMENT Function Signature 4

This function returns a document as BLOB using XML based report data and RTF or XSL-FO based report layout.

Syntax

```
APEX_UTIL.GET_PRINT_DOCUMENT (
    p_report_data         IN CLOB,
    p_report_layout       IN CLOB,
    p_report_layout_type  IN VARCHAR2 default 'xsl-fo',
    p_document_format     IN VARCHAR2 default 'pdf',
    p_print_server        IN VARCHAR2 default NULL)
RETURN BLOB;
```

Parameters

Table 1–46 describes the parameters available in the GET_PRINT_DOCUMENT function. for Signature 4

Table 1–46 GET_PRINT_DOCUMENT Parameters

Parameter	Description
p_report_data	XML based report data, must be encoded in UTF-8
p_report_layout	Report layout in XSL-FO or RTF format
p_report_layout_type	Defines the report layout type, that is "xsl-fo" or "rtf"
p_document_format	Defines the document format, that is "pdf", "rtf", "xls", "htm", or "xml"
p_print_server	URL of the print server. If not specified, the print server will be derived from preferences

Example for Signature 4

The following example shows how to use the GET_PRINT_DOCUMENT using Signature 4 (Document returns as a BLOB using XML based report data and RTF or XSL-FO based report layout). In this example, GET_PRINT_DOCUMENT is used in conjunction with APEX_MAIL.SEND and APEX_MAIL.ADD_ATTACHMENT to send an email with an attachment of the file returned by GET_PRINT_DOCUMENT. Both the report data and layout are taken from values stored in page items (P1_XML and P1_XSL).

```
DECLARE
    l_id number;
    l_document BLOB;
BEGIN
    l_document := APEX_UTIL.GET_PRINT_DOCUMENT (
        p_report_data         => :P1_XML,
        p_report_layout       => :P1_XSL,
        p_report_layout_type  => 'xsl-fo',
        p_document_format     => 'pdf');

   l_id := APEX_MAIL.SEND(
       p_to        => :P35_MAIL_TO,
       p_from      => 'noreplies@oracle.com',
       p_subj      => 'sending PDF via print API',
       p_body      => 'Please review the attachment.',
       p_body_html => 'Please review the attachment');
```

```
   APEX_MAIL.ADD_ATTACHMENT (
       p_mail_id    => l_id,
       p_attachment => l_document,
       p_filename   => 'mydocument.pdf',
       p_mime_type  => 'application/pdf');
END;
```

GET_SCREEN_READER_MODE_TOGGLE Function

This function returns a link to the current page to turn on or off, toggle, the mode. For example, if you are in standard mode, this function displays a link that when clicked switches screen reader mode on.

Syntax

```
APEX_UTIL.GET_SCREEN_READER_MODE_TOGGLE (
    p_on_message  IN VARCHAR2 DEFAULT NULL,
    p_off_message IN VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;
```

Parameters

Table 1–47 describes the parameters available in `GET_SCREEN_READER_MODE_TOGGLE` function.

Table 1–47 GET_SCREEN_READER_MODE_TOGGLE Parameters

Parameter	Description
`p_on_message`	Optional text used for the link to switch to screen reader mode, when you are in standard mode. If this parameter is not passed, the default 'Set Screen Reader Mode On' text will be returned in the link.
`p_off_message`	Optional text used for the link to switch to standard mode, when you are in screen reader mode. If this parameter is not passed, the default 'Set Screen Reader Mode Off' text will be returned in the link.

Example

When running in standard mode, this function will return a link with the text 'Set Screen Reader Mode On'. When the link is clicked the current page is refreshed and screen reader mode is switched on. When running in screen reader mode, a link 'Set Screen Reader Mode Off' is returned. When the link is clicked the current page is refreshed and switched back to standard mode.

```
BEGIN
    htp.p(apex_util.get_screen_reader_mode_toggle);
END;
```

See Also: "SHOW_SCREEN_READER_MODE_TOGGLE Procedure" on page 1-123

GET_SESSION_LANG Function

This function returns the language setting for the current user in the current Application Express session.

Syntax

```
APEX_UTIL.GET_SESSION_LANG
RETURN VARCHAR2;
```

Parameters

None.

Example

The following example shows how to use the GET_SESSION_LANG function. It returns the session language for the current user in the current Application Express session into a local variable.

```
DECLARE
    VAL VARCHAR2(5);
BEGIN
    VAL := APEX_UTIL.GET_SESSION_LANG;
END;
```

GET_SESSION_STATE Function

This function returns the value for an item. You can use this function in your Oracle Application Express applications wherever you can use PL/SQL or SQL. You can also use the shorthand, function V, in place of APEX_UTIL.GET_SESSION_STATE.

Syntax

```
APEX_UTIL.GET_SESSION_STATE (
    p_item    IN   VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 1–48 describes the parameters available in GET_SESSION_STATE function.

Table 1–48 GET_SESSION_STATE Parameters

Parameter	Description
p_item	The case insensitive name of the item for which you want to have the session state fetched

Example

The following example shows how to use the GET_SESSION_STATE function to return the value stored in session state for the item 'my_item'.

```
DECLARE
    l_item_value  VARCHAR2(255);
BEGIN
    l_item_value := APEX_UTIL.GET_SESSION_STATE('my_item');
END;
```

See Also: "GET_NUMERIC_SESSION_STATE Function" on page 1-64 and "SET_SESSION_STATE Procedure" on page 1-119

GET_SESSION_TERRITORY Function

This function returns the territory setting for the current user in the current Application Express session.

Syntax

```
APEX_UTIL.GET_SESSION_TERRITORY
RETURN VARCHAR2;
```

Parameters

None.

Example

The following example shows how to use the `GET_SESSION_TERRITORY` function. It returns the session territory setting for the current user in the current Application Express session into a local variable.

```
DECLARE
    VAL VARCHAR2(30);
BEGIN
    VAL := APEX_UTIL.GET_SESSION_TERRITORY;
END;
```

GET_SESSION_TIME_ZONE Function

This function returns the time zone for the current user in the current Application Express session. This value is null if the time zone is not explicitly set via APEX_UTIL.SET_SESSION_TIME_ZONE or if an application's automatic time zone attribute is enabled.

Syntax

```
APEX_UTIL.GET_SESSION_TIME_ZONE
RETURN VARCHAR2;
```

Parameters

None.

Example

The following example shows how to use the GET_SESSION_TIME_ZONE function. It returns the session time zone for the current user in the current Application Express session into a local variable.

```
BEGIN
    VAL := APEX_UTIL.GET_SESSION_TIME_ZONE;
END;
```

GET_USER_ID Function

This function returns the numeric ID of a named user in the workspace.

Syntax

```
APEX_UTIL.GET_USER_ID(
    p_username   IN VARCHAR2)
RETURN NUMBER;
```

Parameters

Table 1–49 describes the parameters available in `GET_USER_ID` function.

Table 1–49 GET_USER_ID Parameters

Parameter	Description
`p_username`	Identifies the name of a user in the workspace

Example

The following example shows how to use the `GET_USER_ID` function to return the ID for the user named 'FRANK'.

```
DECLARE
    VAL NUMBER;
BEGIN
    VAL := APEX_UTIL.GET_USER_ID(p_username => 'FRANK');
END;
```

GET_USER_ROLES Function

This function returns the DEVELOPER_ROLE field stored in the named user account record. Please note that currently this parameter is named inconsistently between the CREATE_USER, EDIT_USER and FETCH_USER APIs, although they all relate to the DEVELOPER_ROLE field. CREATE_USER uses p_developer_privs, EDIT_USER uses p_developer_roles and FETCH_USER uses p_developer_role.

Syntax

```
APEX_UTIL.GET_USER_ROLES(
   p_username IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 1–50 describes the parameters available in GET_USER_ROLES function.

Table 1–50 GET_USER_ROLES Parameters

Parameter	Description
p_username	Identifies a user name in the account

Example

The following example shows how to use the GET_USER_ROLES function to return colon separated list of roles stored in the DEVELOPER_ROLE field for the user 'FRANK'.

```
DECLARE
    VAL VARCHAR2(4000);
BEGIN
    VAL := APEX_UTIL.GET_USER_ROLES(p_username=>'FRANK');
END;
```

GET_USERNAME Function

This function returns the user name of a user account identified by a numeric ID.

Syntax

```
APEX_UTIL.GET_USERNAME(
    p_userid IN NUMBER)
RETURN VARCHAR2;
```

Parameters

Table 1–51 describes the parameters available in GET_USERNAME function.

Table 1–51 GET_USERNAME Parameters

Parameter	Description
p_userid	Identifies the numeric ID of a user account in the workspace

Example

The following example shows how to use the GET_USERNAME function to return the user name for the user with an ID of 228922003.

```
DECLARE
    VAL VARCHAR2(100);
BEGIN
    VAL := APEX_UTIL.GET_USERNAME(p_userid => 228922003);
END;
```

See Also: "SET_USERNAME Procedure" on page 1-122

IR_CLEAR Procedure

This procedure clears report settings.

Syntax

```
APEX_UTIL.IR_CLEAR(
    p_page_id IN NUMBER,
    p_report_alias IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 1–52 describes the parameters available in `IR_CLEAR` procedure.

Table 1–52 IR_CLEAR Parameters

Parameter	Description
`p_page_id`	Page of the current APEX application that contains an interactive report.
`p_report_alias`	To clear a primary report, leave `p_report_alias` as null.

IR_FILTER Procedure

This procedure creates a report filter.

Syntax

```
APEX_UTIL.IR_FILTER(
    p_page_id       IN NUMBER,
    p_report_column IN VARCHAR2,
    p_operator_abbr IN VARCHAR2 DEFAULT NULL,
    p_filter_value  IN VARCHAR2,
    p_report_alias  IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 1–53 describes the parameters available in `IR_FILTER` procedure.

Table 1–53 IR_FILTER Parameters

Parameter	Description
`p_page_id`	Page of the current APEX application that contains an interactive report.
`p_report_column`	Name of the report SQL column, or column alias, to filtered.
`p_operator_abbr`	Filter type. Valid values are as follows: ■ EQ = Equals ■ NEQ = Not Equals ■ LT = Less than ■ LTE = Less then or equal to ■ GT = Greater Than ■ GTE = Greater than or equal to ■ LIKE = SQL Like operator ■ N = Null ■ NN = Not Null ■ C = Contains ■ NC = Not Contains ■ IN = SQL In Operator ■ NIN = SQL Not In Operator
p_filter_value	Filter value. This value is not used for 'N' and 'NN'.
`p_report_alias`	Identifies the saved report alias within the current application page.

IR_RESET Procedure

This procedure resets report settings.

Syntax

```
APEX_UTIL.IR_RESET(
    p_page_id IN NUMBER,
    p_report_alias IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 1–54 describes the parameters available in `IR_RESET` procedure.

Table 1–54 IR_RESET Parameters

Parameter	Description
`p_page_id`	Page of the current APEX application that contains an interactive report.
`p_report_alias`	To clear a primary report, leave `p_report_alias` as null.

IS_LOGIN_PASSWORD_VALID Function

This function returns a Boolean result based on the validity of the password for a named user account in the current workspace. This function returns true if the password matches and it returns false if the password does not match.

Syntax

```
APEX_UTIL.IS_LOGIN_PASSWORD_VALID(
    p_username IN VARCHAR2,
    p_password IN VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 1–55 describes the parameters available in the `IS_LOGIN_PASSWORD_VALID` function.

Table 1–55 IS_LOGIN_PASSWORD_VALID Parameters

Parameter	Description
`p_username`	User name in account
`p_password`	Password to be compared with password stored in the account

Example

The following example shows how to use the `IS_LOGIN_PASSWORD_VALID` function to check if the user 'FRANK' has the password 'tiger'. `TRUE` will be returned if this is a valid password for 'FRANK', `FALSE` if not.

```
DECLARE
    VAL BOOLEAN;
BEGIN
    VAL := APEX_UTIL.IS_LOGIN_PASSWORD_VALID (
        p_username=>'FRANK',
        p_password=>'tiger');
END;
```

IS_SCREEN_READER_SESSION Function

This function returns a boolean true if the session is in screen reader mode and returns a boolean false if not in screen reader mode.

Syntax

```
APEX_UTIL.IS_SCREEN_READER_SESSION
RETURN BOOLEAN;
```

Parameters

None

Example

```
BEGIN
    IF apex_util.is_screen_reader_session then
        htp.p('Screen Reader Mode');
    END IF;
END;
```

IS_SCREEN_READER_SESSION_YN Function

This function returns 'Y' if the session is in screen reader mode and 'N' if not in screen reader mode.

Syntax

```
APEX_UTIL.IS_SCREEN_READER_SESSION_YN
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
    IF apex_util.is_screen_reader_session_yn = 'Y' then
        htp.p('Screen Reader Mode');
    END IF;
END;
```

IS_USERNAME_UNIQUE Function

This function returns a Boolean result based on whether the named user account is unique in the workspace.

Syntax

```
APEX_UTIL.IS_USERNAME_UNIQUE(
    p_username IN VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 1–56 describes the parameters available in `IS_USERNAME_UNIQUE` function.

Table 1–56 IS_USERNAME_UNIQUE Parameters

Parameter	Description
`p_username`	Identifies the user name to be tested

Example

The following example shows how to use the `IS_USERNAME_UNIQUE` function. If the user 'FRANK' already exists in the current workspace, `FALSE` will be returned, otherwise `TRUE` is returned.

```
DECLARE
    VAL BOOLEAN;
BEGIN
    VAL := APEX_UTIL.IS_USERNAME_UNIQUE(
        p_username=>'FRANK');
END;
```

KEYVAL_NUM Function

This function gets the value of the package variable (`wwv_flow_utilities.g_val_num`) set by `APEX_UTIL.SAVEKEY_NUM`.

Syntax

```
APEX_UTIL.KEYVAL_NUM
RETURN NUMBER;
```

Parameters

None

Example

The following example shows how to use the `KEYVAL_NUM` function to return the current value of the package variable `wwv_flow_utilities.g_val_num`.

```
DECLARE
    VAL NUMBER;
BEGIN
    VAL := APEX_UTIL.KEYVAL_NUM;
END;
```

See Also: "SAVEKEY_NUM Function" on page 1-101

KEYVAL_VC2 Function

This function gets the value of the package variable (`wwv_flow_utilities.g_val_vc2`) set by `APEX_UTIL.SAVEKEY_VC2`.

Syntax

```
APEX_UTIL.KEYVAL_VC2;
```

Parameters

None.

Example

The following example shows how to use the `KEYVAL_VC2` function to return the current value of the package variable `wwv_flow_utilities.g_val_vc2`.

```
DECLARE
    VAL VARCHAR2(4000);
BEGIN
    VAL := APEX_UTIL.KEYVAL_VC2;
END;
```

See Also: "SAVEKEY_VC2 Function" on page 1-102

LOCK_ACCOUNT Procedure

Sets a user account status to locked. Must be run by an authenticated workspace administrator in the context of a page request.

Syntax

```
APEX_UTIL.LOCK_ACCOUNT (
     p_user_name IN VARCHAR2);
```

Parameters

Table 1–57 describes the parameters available in the LOCK_ACCOUNT procedure.

Table 1–57 LOCK_ACCOUNT Parameters

Parameter	Description
p_user_name	The user name of the user account

Example

The following example shows how to use the LOCK_ACCOUNT procedure. Use this procedure to lock an Application Express account (workspace administrator, developer, or end user) in the current workspace. This action locks the account for use by administrators, developers, and end users.

```
BEGIN
    FOR c1 IN (SELECT user_name from wwv_flow_users) LOOP
        APEX_UTIL.LOCK_ACCOUNT(p_user_name => c1.user_name);
        htp.p('End User Account:'||c1.user_name||' is now locked.');
    END LOOP;
END;
```

See Also: "UNLOCK_ACCOUNT Procedure" on page 1-136 and "GET_ACCOUNT_LOCKED_STATUS Function" on page 1-46

PASSWORD_FIRST_USE_OCCURRED Function

Returns true if the account's password has changed since the account was created, an Oracle Application Express administrator performs a password reset operation that results in a new password being emailed to the account holder, or a user has initiated password reset operation. This function returns false if the account's password has not been changed since either of the events just described.

This function may be run in a page request context by any authenticated user.

Syntax

```
APEX_UTIL.PASSWORD_FIRST_USE_OCCURRED (
    p_user_name IN VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 1–58 describes the parameters available in the PASSWORD_FIRST_USE_OCCURRED procedure.

Table 1–58 PASSWORD_FIRST_USE_OCCURRED Parameters

Parameter	Description
p_user_name	The user name of the user account

Example

The following example shows how to use the PASSWORD_FIRST_USE_OCCURRED function. Use this function to check if the password for an Application Express user account (workspace administrator, developer, or end user) in the current workspace has been changed by the user the first time the user logged in after the password was initially set during account creation, or was changed by one of the password reset operations described above.

This is meaningful only with accounts for which the CHANGE_PASSWORD_ON_FIRST_USE attribute is set to **Yes**.

```
BEGIN
    FOR c1 IN (SELECT user_name from wwv_flow_users) LOOP
        IF APEX_UTIL.PASSWORD_FIRST_USE_OCCURRED(p_user_name => c1.user_name) THEN
            htp.p('User:'||c1.user_name||' has logged in and updated the
password.');
        END IF;
    END LOOP;
END;
```

See Also: "CHANGE_PASSWORD_ON_FIRST_USE Function" on page 1-11

PREPARE_URL Function

The PREPARE_URL function serves two purposes:

1. To return an f?p URL with the Session State Protection checksum argument (&cs=) if one is required.
2. To return an f?p URL with the session ID component replaced with zero (0) if the zero session ID feature is in use and other criteria are met.

> **Note:** The `PREPARE_URL` functions returns the f?p URL with `&cs=<large hex value>` appended. If you use this returned value, for example in JavaScript, it may be necessary to escape the ampersand in the URL in order to conform with syntax rules of the particular context. One place you may encounter this is in SVG chart SQL queries which might include `PREPARE_URL` calls.

Syntax

```
APEX_UTIL.PREPARE_URL (
    p_url           IN VARCHAR2,
    p_url_charset   IN VARCHAR2 default null,
    p_checksum_type IN VARCHAR2 default null)
RETURN VARCHAR2;
```

Parameters

Table 1–59 describes the parameters available in the PREPARE_URL function.

Table 1–59 PREPARE_URL Parameters

Parameter	Description
p_url	An f?p relative URL with all substitutions resolved
p_url_charset	The character set name (for example, `UTF-8`) to use when escaping special characters contained within argument values
p_checksum type	Null or any of the following six values, `SESSION` or `3`, `PRIVATE_BOOKMARK` or `2`, or `PUBLIC_BOOKMARK` or `1`

Example 1

The following example shows how to use the `PREPARE_URL` function to return a URL with a valid 'SESSION' level checksum argument. This URL sets the value of `P1_ITEM` page item to `xyz`.

```
DECLARE
    l_url varchar2(2000);
    l_app number := v('APP_ID');
    l_session number := v('APP_SESSION');
BEGIN
    l_url := APEX_UTIL.PREPARE_URL(
        p_url => 'f?p=' || l_app || ':1:'||l_session||'::NO::P1_ITEM:xyz',
        p_checksum_type => 'SESSION');
END;
```

Example 2

The following example shows how to use the PREPARE_URL function to return a URL with a zero session ID. In a PL/SQL Dynamic Content region that generates f?p URLs (anchors), call PREPARE_URL to ensure that the session ID will set to zero when the zero session ID feature is in use, when the user is a public user (not authenticated), and when the target page is a public page in the current application:

```
htp.p(APEX_UTIL.PREPARE_URL(p_url => 'f?p=' || :APP_ID || ':10:'|| :APP_SESSION
||'::NO::P10_ITEM:ABC');
```

When using PREPARE_URL for this purpose, the p_url_charset and p_checksum_type arguments can be omitted. However, it is permissible to use them when both the Session State Protection and Zero Session ID features are applicable.

> **See Also:** "Facilitating Bookmarks by Using Zero as the Session ID" and "Understanding Session State Protection" in *Oracle Application Express Advanced Tutorials*

PUBLIC_CHECK_AUTHORIZATION Function

Given the name of a security scheme, this function determines if the current user passes the security check.

Syntax

```
APEX_UTIL.PUBLIC_CHECK_AUTHORIZATION (
    p_security_scheme    IN    VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 1–60 describes the parameters available in `PUBLIC_CHECK_AUTHORIZATION` function.

Table 1–60 PUBLIC_CHECK_AUTHORIZATION Parameters

Parameter	Description
`p_security_name`	The name of the security scheme that determines if the user passes the security check

Example

The following example shows how to use the `PUBLIC_CHECK_AUTHORIZATION` function to check if the current user passes the check defined in the `my_auth_scheme` authorization scheme.

```
DECLARE
    l_check_security  BOOLEAN;
BEGIN
    l_check_security := APEX_UTIL.PUBLIC_CHECK_AUTHORIZATION('my_auth_scheme');
END;
```

PURGE_REGIONS_BY_APP Procedure

Deletes all cached regions for an application.

Syntax

```
APEX_UTIL.PURGE_REGIONS_BY_APP (
    p_application IN NUMBER);
```

Parameters

Table 1–61 describes the parameters available in PURGE_REGIONS_BY_APP.

Table 1–61 PURGE_REGIONS_BY_APP Parameters

Parameter	Description
p_application	The identification number (ID) of the application.

Example

The following example show how to use APEX_UTIL.PURGE_REGIONS_BY_APP to delete all cached regions for application #123.

```
BEGIN
    APEX_UTILITIES.PURGE_REGIONS_BY_APP(p_application=>123);
END;
```

PURGE_REGIONS_BY_NAME Procedure

Deletes all cached values for a region identified by the application ID, page number and region name.

Syntax

```
APEX_UTIL.PURGE_REGIONS_BY_NAME (
     p_application IN NUMBER,
     p_page        IN NUMBER,
     p_region_name IN VARCHAR2);
```

Parameters

Table 1–62 describes the parameters available in PURGE_REGIONS_BY_NAME.

Table 1–62 PURGE_REGIONS_BY_NAME Parameters

Parameter	Description
p_application	The identification number (ID) of the application.
p_page	The number of the page containing the region to be deleted.
p_region_name	The region name to be deleted.

Example

The following example shows how to use the PURGE_REGIONS_BY_NAME procedure to delete all the cached values for the region 'my_cached_region' on page 1 of the current application.

```
BEGIN
    APEX_UTIL.PURGE_REGIONS_BY_NAME(
        p_application => :APP_ID,
        p_page => 1,
        p_region_name => 'my_cached_region');
END;
```

PURGE_REGIONS_BY_PAGE Procedure

Deletes all cached regions by application and page.

Syntax

```
APEX_UTIL.PURGE_REGIONS_BY_PAGE (
     p_application IN NUMBER,
     p_page     IN NUMBER);
```

Parameters

Table 1–63 describes the parameters available in PURGE_REGIONS_BY_PAGE.

Table 1–63 PURGE_REGIONS_BY_PAGE Parameters

Parameter	Description
p_application	The identification number (ID) of the application.
p_page	The identification number of page containing the region.

Example

The following example shows how to use the PURGE_REGIONS_BY_PAGE procedure to delete all the cached values for regions on page 1 of the current application.

```
BEGIN
    APEX_UTIL.PURGE_REGIONS_BY_PAGE(
        p_application => :APP_ID,
        p_page => 1);
END;
```

REMOVE_PREFERENCE Procedure

This procedure removes the preference for the supplied user.

Syntax

```
APEX_UTIL.REMOVE_PREFERENCE(
    p_preference    IN    VARCHAR2 DEFAULT NULL,
    p_user          IN    VARCHAR2 DEFAULT V('USER'));
```

Parameters

Table 1–64 describes the parameters available in the `REMOVE_PREFERENCE` procedure.

Table 1–64 REMOVE_PREFERENCE Parameters

Parameter	Description
`p_preference`	Name of the preference to remove
`p_user`	User for whom the preference is defined

Example

The following example shows how to use the `REMOVE_PREFERENCE` procedure to remove the preference `default_view` for the currently authenticated user.

```
BEGIN
    APEX_UTIL.REMOVE_PREFERENCE(
        p_preference => 'default_view',
        p_user       => :APP_USER);
END;
```

See Also: "GET_PREFERENCE Function" on page 1-65, "SET_PREFERENCE Procedure" on page 1-110 and "Managing Session State and User Preferences" in *Oracle Application Express Administration Guide*.

REMOVE_SORT_PREFERENCES Procedure

This procedure removes the user's column heading sorting preference value.

Syntax

```
APEX_UTIL.REMOVE_SORT_PREFERENCES (
    p_user  IN   VARCHAR2 DEFAULT V('USER'));
```

Parameters

Table 1–65 describes the parameters available in REMOVE_SORT_PREFERENCES function.

Table 1–65 REMOVE_SORT_PREFERENCES Parameters

Parameter	Description
p_user	Identifies the user for whom sorting preferences will be removed

Example

The following example shows how to use the REMOVE_SORT_PREFERENCES procedure to remove the currently authenticated user's column heading sorting preferences.

```
BEGIN
    APEX_UTIL.REMOVE_SORT_PREFERENCES(:APP_USER);
END;
```

REMOVE_USER Procedure

This procedure removes the user account identified by the primary key or a user name. To execute this procedure, the current user must have administrative privilege in the workspace.

Syntax

```
APEX_UTIL.REMOVE_USER(
    p_user_id   IN NUMBER,
    p_user_name IN VARCHAR2);
```

Parameters

Table 1–66 describes the parameters available in the REMOVE_USER procedure.

Table 1–66 REMOVE_USER Parameters

Parameter	Description
p_user_id	The numeric primary key of the user account record
p_user_name	The user name of the user account

Example

The following examples show how to use the REMOVE_USER procedure to remove a user account. Firstly, by the primary key (using the p_user_id parameter) and secondly by user name (using the p_user_name parameter).

```
BEGIN
    APEX_UTIL.REMOVE_USER(p_user_id=> 99997);
END;

BEGIN
    APEX_UTIL.REMOVE_USER(p_user_name => 'FRANK');
END;
```

RESET_AUTHORIZATIONS Procedure

To increase performance, Oracle Application Express caches the results of authorization schemes after they have been evaluated. You can use this procedure to undo caching, requiring each authorization scheme be revalidated when it is next encountered during page show or accept processing. You can use this procedure if you want users to have the ability to change their responsibilities (their authorization profile) within your application.

Syntax

```
APEX_UTIL.RESET_AUTHORIZATIONS;
```

Parameters

None.

Example

The following example shows how to use the `RESET_AUTHORIZATIONS` procedure to clear the authorization scheme cache.

```
BEGIN
    APEX_UTIL.RESET_AUTHORIZATIONS;
END;
```

RESET_PW Procedure

This procedure resets the password for a named user and emails it in a message to the email address located for the named account in the current workspace. To execute this procedure, the current user must have administrative privilege in the workspace.

Syntax

```
APEX_UTIL.RESET_PW(
    p_user IN VARCHAR2,
    p_msg  IN VARCHAR2);
```

Parameters

Table 1–67 describes the parameters available in the `RESET_PW` procedure.

Table 1–67 RESET_PW Parameters

Parameter	Description
`p_user`	The user name of the user account
`p_msg`	Message text to be mailed to a user

Example

The following example shows how to use the `RESET_PW` procedure to reset the password for the user 'FRANK'.

```
BEGIN
    APEX_UTIL.RESET_PW(
        p_user => 'FRANK',
        p_msg => 'Contact help desk at 555-1212 with questions');
END;
```

See Also: "CHANGE_CURRENT_USER_PW Procedure" on page 1-10

SAVEKEY_NUM Function

This function sets a package variable (wwv_flow_utilities.g_val_num) so that it can be retrieved using the function KEYVAL_NUM.

Syntax

```
APEX_UTIL.SAVEKEY_NUM(
    p_val IN NUMBER)
RETURN NUMBER;
```

Parameters

Table 1–68 describes the parameters available in the SAVEKEY_NUM procedure.

Table 1–68 SAVEKEY_NUM Parameters

Parameter	Description
p_val	The numeric value to be saved

Example

The following example shows how to use the SAVEKEY_NUM function to set the wwv_flow_utilities.g_val_num package variable to the value of 10.

```
DECLARE
    VAL NUMBER;
BEGIN
    VAL := APEX_UTIL.SAVEKEY_NUM(p_val => 10);
END;
```

See Also: "KEYVAL_NUM Function" on page 1-86

SAVEKEY_VC2 Function

This function sets a package variable (`wwv_flow_utilities.g_val_vc2`) so that it can be retrieved using the function `KEYVAL_VC2`.

Syntax

```
APEX_UTIL.SAVEKEY_VC2(
    p_val IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 1–69 describes the parameters available in the `SAVEKEY_VC2` function.

Table 1–69 SAVEKEY_VC2 Parameters

Parameter	Description
`p_val`	The is the VARCHAR2 value to be saved

Example

The following example shows how to use the `SAVEKEY_VC2` function to set the `wwv_flow_utilities.g_val_vc2` package variable to the value of 'XXX'.

```
DECLARE
    VAL VARCHAR2(4000);
BEGIN
    VAL := APEX_UTIL.SAVEKEY_VC2(p_val => 'XXX');
END;
```

See Also: "KEYVAL_VC2 Function" on page 1-87

SET_ATTRIBUTE Procedure

This procedure sets the value of one of the attribute values (1 through 10) of a user in the Application Express accounts table.

Syntax

```
APEX_UTIL.SET_ATTRIBUTE(
    p_userid           IN NUMBER,
    p_attribute_number IN NUMBER,
    p_attribute_value  IN VARCHAR2);
```

Parameters

Table 1–70 describes the parameters available in the SET_ATTRIBUTE procedure.

Table 1–70 SET_ATTRIBUTE Parameters

Parameter	Description
p_userid	The numeric ID of the user account
p_attribute_number	Attribute number in the user record (1 through 10)
p_attribute_value	Value of the attribute located by p_attribute_number to be set in the user record

Example

The following example shows how to use the SET_ATTRIBUTE procedure to set the number 1 attribute for user 'FRANK' with the value 'foo'.

```
DECLARE
    VAL VARCHAR2(4000);
BEGIN
    APEX_UTIL.SET_ATTRIBUTE (
        p_userid => apex_util.get_user_id(p_username => 'FRANK'),
        p_attribute_number => 1,
        p_attribute_value => 'foo');
END;
```

See Also: "GET_ATTRIBUTE Function" on page 1-47

SET_AUTHENTICATION_RESULT Procedure

This procedure can be called from an application's custom authentication function (that is, credentials verification function). The status passed to this procedure is logged in the Login Access Log.

> **See Also:** "Monitoring Activity within a Workspace" in *Oracle Application Express Administration Guide*

Syntax

```
APEX_UTIL.SET_AUTHENTICATION_RESULT(
    p_code IN NUMBER);
```

Parameters

Table 1–21 describes the parameters available in the `SET_AUTHENTICATION_RESULT` procedure.

Table 1–71 SET_AUTHENTICATION_RESULT Parameters

Parameter	Description
`p_code`	Any numeric value the developer chooses. After this value is set in the session using this procedure, it can be retrieved using the `APEX_UTIL.GET_AUTHENTICATION_RESULT` function.

Example

One way to use this procedure is to include it in the application authentication scheme. This example demonstrates how text and numeric status values can be registered for logging. In this example, no credentials verification is performed, it just demonstrates how text and numeric status values can be registered for logging.

Note that the status set using this procedure is visible in the `apex_user_access_log` view and in the reports on this view available to workspace and site administrators.

```
CREATE OR REPLACE FUNCTION MY_AUTH(
    p_username IN VARCHAR2,
    p_password IN VARCHAR2)
RETURN BOOLEAN
IS
BEGIN
    APEX_UTIL.SET_CUSTOM_AUTH_STATUS(p_status=>'User:'||p_username||' is back.');
    IF UPPER(p_username) = 'GOOD' THEN
        APEX_UTIL.SET_AUTHENTICATION_RESULT(24567);
        RETURN TRUE;
    ELSE
        APEX_UTIL.SET_AUTHENTICATION_RESULT(-666);
        RETURN FALSE;
    END IF;
END;
```

> **See Also:** "GET_AUTHENTICATION_RESULT Function" on page 1-48 and "SET_CUSTOM_AUTH_STATUS Procedure" on page 1-105

SET_CUSTOM_AUTH_STATUS Procedure

This procedure can be called from an application's custom authentication function (that is, credentials verification function). The status passed to this procedure is logged in the Login Access Log.

> **See Also:** "Monitoring Activity within a Workspace" in *Oracle Application Express Administration Guide*

Syntax

```
APEX_UTIL.SET_CUSTOM_AUTH_STATUS(
    p_status  IN VARCHAR2);
```

Parameters

Table 1–72 describes the parameters available in the SET_CUSTOM_AUTH_STATUS procedure.

Table 1–72 SET_CUSTOM_AUTH_STATUS Parameters

Parameter	Description
p_status	Any text the developer chooses to denote the result of the authentication attempt (up to 4000 characters).

Example

One way to use the SET_CUSTOM_AUTH_STATUS procedure is to include it in the application authentication scheme. This example demonstrates how text and numeric status values can be registered for logging. Note that no credentials verification is performed.

The status set using this procedure is visible in the apex_user_access_log view and in the reports on this view available to workspace and site administrators.

```
CREATE OR REPLACE FUNCTION MY_AUTH(
    p_username IN VARCHAR2,
    p_password IN VARCHAR2)
RETURN BOOLEAN
IS
BEGIN
    APEX_UTIL.SET_CUSTOM_AUTH_STATUS(p_status=>'User:'||p_username||' is back.');
    IF UPPER(p_username) = 'GOOD' THEN
        APEX_UTIL.SET_AUTHENTICATION_RESULT(24567);
        RETURN TRUE;
    ELSE
        APEX_UTIL.SET_AUTHENTICATION_RESULT(-666);
        RETURN FALSE;
    END IF;
END;
```

> **See Also:** "SET_AUTHENTICATION_RESULT Procedure" on page 1-104 and "GET_AUTHENTICATION_RESULT Function" on page 1-48

SET_EDITION Procedure

This procedure sets the name of the edition to be used in all application SQL parsed in the current page view or page submission.

Syntax

```
APEX_UTIL.SET_EDITION(
    p_edition IN VARCHAR2);
```

Parameters

Table 1–72 describes the parameters available in the `SET_EDITION` procedure.

Table 1–73 SET_EDITION Parameters

Parameter	Description
`p_edition`	Edition name.

Example

The following example shows how to use the SET_EDITION procedure. It sets the edition name for the database session of the current page view.

```
BEGIN
    APEX_UTIL.SET_EDITION( P_EDITION => 'Edition1' );
END;
```

Note: Support for Edition-Based Redefinition is only available in database version 11.2.0.1 or higher.

SET_EMAIL Procedure

This procedure updates a user account with a new email address. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
APEX_UTIL.SET_EMAIL(
    p_userid IN NUMBER,
    p_email  IN VARCHAR2);
```

Parameters

Table 1–74 describes the parameters available in the SET_EMAIL procedure.

Table 1–74 SET_EMAIL Parameters

Parameter	Description
p_userid	The numeric ID of the user account
p_email	The email address to be saved in user account

Example

The following example shows how to use the SET_EMAIL procedure to set the value of EMAIL to 'frank.scott@somewhere.com' for the user 'FRANK'.

```
BEGIN
    APEX_UTIL.SET_EMAIL(
        p_userid  => APEX_UTIL.GET_USER_ID('FRANK'),
        p_email   => 'frank.scott@somewhere.com');
END;
```

See Also: "GET_EMAIL Function" on page 1-54 and "GET_USER_ID Function" on page 1-76

SET_FIRST_NAME Procedure

This procedure updates a user account with a new FIRST_NAME value. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
APEX_UTIL.SET_FIRST_NAME(
    p_userid      IN NUMBER,
    p_first_name  IN VARCHAR2);
```

Parameters

Table 1–75 describes the parameters available in the SET_FIRST_NAME procedure.

Table 1–75 SET_FIRST_NAME Parameters

Parameter	Description
p_userid	The numeric ID of the user account
p_first_name	FIRST_NAME value to be saved in user account

Example

The following example shows how to use the SET_FIRST_NAME procedure to set the value of FIRST_NAME to 'FRANK' for the user 'FRANK'.

```
BEGIN
    APEX_UTIL.SET_FIRST_NAME(
        p_userid       => APEX_UTIL.GET_USER_ID('FRANK'),
        p_first_name   => 'FRANK');
END;
```

See Also: "GET_FIRST_NAME Function" on page 1-59and "GET_USER_ID Function" on page 1-76

SET_LAST_NAME Procedure

This procedure updates a user account with a new LAST_NAME value. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
APEX_UTIL.SET_LAST_NAME(
    p_userid      IN NUMBER,
    p_last_name   IN VARCHAR2);
```

Parameters

Table 1–76 describes the parameters available in the SET_LAST_NAME procedure.

Table 1–76 SET_LAST_NAME Parameters

Parameter	Description
p_userid	The numeric ID of the user account
p_last_name	LAST_NAME value to be saved in the user account

Example

The following example shows how to use the SET_LAST_NAME procedure to set the value of LAST_NAME to 'SMITH' for the user 'FRANK'.

```
BEGIN
    APEX_UTIL.SET_LAST_NAME(
        p_userid       => APEX_UTIL.GET_USER_ID('FRANK'),
        p_last_name    => 'SMITH');
END;
```

See Also: "GET_LAST_NAME Function" on page 1-63 and "GET_USER_ID Function" on page 1-76

SET_PREFERENCE Procedure

This procedure sets a preference that will persist beyond the user's current session.

Syntax

```
APEX_UTIL.SET_PREFERENCE (
    p_preference   IN    VARCHAR2 DEFAULT NULL,
    p_value        IN    VARCHAR2 DEFAULT NULL,
    p_user         IN    VARCHAR2 DEFAULT NULL);
```

Parameters

Table 1–77 describes the parameters available in the `SET_PREFERENCE` procedure.

Table 1–77 SET_PREFERENCE Parameters

Parameter	Description
`p_preference`	Name of the preference (case-sensitive)
`p_value`	Value of the preference
`p_user`	User for whom the preference is being set

Example

The following example shows how to use the `SET_PREFERENCE` procedure to set a preference called '`default_view`' to the value 'WEEKLY' that will persist beyond session for the currently authenticated user.

```
BEGIN
    APEX_UTIL.SET_PREFERENCE(
        p_preference => 'default_view',
        p_value      => 'WEEKLY',
        p_user       => :APP_USER);
END;
```

See Also: "GET_PREFERENCE Function" on page 1-65 and "REMOVE_PREFERENCE Procedure" on page 1-96

SET_SECURITY_GROUP_ID Procedure

This procedure can be used with `apex_util.find_security_group_id` and is used to ease the use of the mail package in batch mode. For example, you might want to create a procedure that is run by a nightly job to email all outstanding tasks.

Syntax

```
APEX_UTIL.SET_SECURITY_GROUP_ID (
    p_security_group_id  IN NUMBER);
```

Parameters

Table 1–78 describes the parameters available in the `SET_SECURITY_GROUP_ID` procedure.

Table 1–78 SET_SECURITY_GROUP_ID Parameters

Parameter	Description
`p_security_group_id`	This is the security group id of the workspace you are working in.

SET_SESSION_LANG Procedure

This procedure sets the language to be used for the current user in the current Application Express session. The language must be a valid IANA language name.

Syntax

```
APEX_UTIL.SET_SESSION_LANG(
    p_lang IN VARCHAR2);
```

Parameters

Table 1–79 describes the parameters available in the SET_SESSION_LANG procedure.

Table 1–79 SET_SESSION_LANG Parameters

Parameter	Description
p_lang	This is an IANA language code. Some examples include: en, de, de-at, zh-cn, and pt-br.

Example

The following example shows how to use the SET_SESSION_LANG procedure. It sets the language for the current user for the duration of the Application Express session.

```
BEGIN
    APEX_UTIL.SET_SESSION_LANG( P_LANG => 'en');
END;
```

SET_SESSION_LIFETIME_SECONDS Procedure

This procedure sets the current application's Maximum Session Length in Seconds value for the current session, overriding the corresponding application attribute. This allows developers to dynamically shorten or lengthen the session life based on criteria determined after the user authenticates.

> **Note:** In order for this procedure to have any effect, the application's Maximum Session Length in Seconds attribute must have been set to a non-zero value in the application definition. This procedure will have no effect if that attribute was not set by the developer.

Syntax

```
APEX_UTIL.SET_SESSION_LIFETIME_SECONDS (
    p_seconds  IN    NUMEBER,
    p_scope    IN    VARCHAR2 DEFAULT 'SESSION');
```

Parameters

Table 1–80 describes the parameters available in the `SET_SESSION_LIFETIME_SECONDS` procedure.

Table 1–80 SET_SESSION_LIFETIME_SECONDS Parameters

Parameter	Description
`p_seconds`	A positive integer indicating the number of seconds the session used by this application is allowed to exist.
`p_scope`	Defaults to 'SESSION' and may also be set to 'APPLICATION'. If 'SESSION', all applications using this session are affected. If 'APPLICATION', only the current application using the current session is affected.

Example 1

The following example shows how to use the SET_SESSION_LIFETIME_SECONDS procedure to set the current application's Maximum Session Length in Seconds attribute to 7200 seconds (two hours). This API call will have no effect if the application's Maximum Session Length in Seconds attribute was not set by the developer to a non-zero value in the application definition.

By allowing the p_scope input parameter to use the default value of 'SESSION', the following example would actually apply to all applications using the current session. This would be the most common use case when multiple Application Express applications use a common authentication scheme and are designed to operate as a suite in a common session.

```
BEGIN
   APEX_UTIL.SET_SESSION_LIFETIME_SECONDS(p_seconds => 7200);
END;
```

Example 2

The following example shows how to use the SET_SESSION_LIFETIME_SECONDS procedure to set the current application's Maximum Session Length in Seconds attribute to 3600 seconds (one hour). This API call will have no effect if the application's Maximum Session Length in Seconds attribute was not set by the developer to a non-zero value in the application definition.

By overriding the p_scope input parameter's default value and setting it to 'APPLICATION', the following example would actually apply to only to the current application using the current session even if other applications are using the same session.

```
BEGIN
    APEX_UTIL.SET_SESSION_LIFETIME_SECONDS(p_seconds => 3600,
    p_scope => 'APPLICATION');
END;
```

SET_SESSION_MAX_IDLE_SECONDS Procedure

Sets the current application's Maximum Session Idle Time in Seconds value for the current session, overriding the corresponding application attribute. This allows developers to dynamically shorten or lengthen the maximum idle time allowed between page requests based on criteria determined after the user authenticates.

> **Note:** In order for this procedure to have any effect, the application's Maximum Session Idle Time in Seconds attribute must have been set to a non-zero value in the application definition. This procedure will have no effect if that attribute was not set by the developer.

Syntax

```
APEX_UTIL.SET_SESSION_MAX_IDLE_SECONDS (
    p_seconds  IN    NUMEBER,
    p_scope    IN    VARCHAR2 DEFAULT 'SESSION');
```

Parameters

Table 1–81 describes the parameters available in the SET_SESSION_MAX_IDLE_SECONDS procedure.

Table 1–81 SET_SESSION_MAX_IDLE_SECONDS Parameters

Parameter	Description
p_seconds	A positive integer indicating the number of seconds allowed between page requests.
p_scope	Defaults to 'SESSION' and may also be set to 'APPLICATION'. If 'SESSION', this idle time applies to all applications using this session. If 'APPLICATION', this idle time only applies to the current application using the current session.

Example 1

The following example shows how to use the SET_SESSION_MAX_IDLE_SECONDS procedure to set the current application's Maximum Session Idle Time in Seconds attribute to 1200 seconds (twenty minutes). This API call will have no effect if the application's Maximum Session Idle Time in Seconds attribute was not set by the developer to a non-zero value in the application definition.

By allowing the p_scope input parameter to use the default value of 'SESSION', the following example would actually apply to all applications using the current session. This would be the most common use case when multiple Application Express applications use a common authentication scheme and are designed to operate as a suite in a common session.

```
BEGIN
   APEX_UTIL.SET_SESSION_MAX_IDLE_SECONDS(p_seconds => 1200);
END;
```

Example 2

The following example shows how to use the SET_SESSION_MAX_IDLE_SECONDS procedure to set the current application's Maximum Session Idle Time in Seconds

attribute to 600 seconds (ten minutes). This API call will have no effect if the application's Maximum Session Idle Time in Seconds attribute was not set by the developer to a non-zero value in the application definition.

By overriding the `p_scope` input parameter's default value and setting it to '`APPLICATION`', the following example would actually apply to only to the current application using the current session even if other applications are using the same session.

```
BEGIN
    APEX_UTIL.SET_SESSION_MAX_IDLE_SECONDS(p_seconds => 600,
    p_scope => 'APPLICATION');
END;
```

SET_SESSION_SCREEN_READER_OFF Procedure

This procedure puts the current session into standard mode.

Syntax

```
APEX_UTIL.SET_SESSION_SCREEN_READER_OFF;
```

Parameters

None

Example

In this example, the current session is put into standard mode.

```
BEGIN
    IF apex_util.set_session_screen_reader_off;
END;
```

SET_SESSION_SCREEN_READER_ON Procedure

This procedure puts the current session into screen reader mode.

Syntax

```
APEX_UTIL.SET_SESSION_SCREEN_READER_ON;
```

Parameters

None

Example

In this example, the current session is put into screen reader mode.

```
BEGIN
    IF apex_util.set_session_screen_reader_on;
END;
```

SET_SESSION_STATE Procedure

This procedure sets session state for a current Oracle Application Express session.

Syntax

```
APEX_UTIL.SET_SESSION_STATE (
    p_name     IN    VARCHAR2 DEFAULT NULL,
    p_value    IN    VARCHAR2 DEFAULT NULL);
```

Parameters

Table 1–82 describes the parameters available in the SET_SESSION_STATE procedure.

Table 1–82 SET_SESSION_STATE Parameters

Parameter	Description
p_name	Name of the application-level or page-level item for which you are setting sessions state
p_value	Value of session state to set

Example

The following example shows how to use the SET_SESSION_STATE procedure to set the value of the item 'my_item' to 'myvalue' in the current session.

```
BEGIN
    APEX_UTIL.SET_SESSION_STATE('my_item','myvalue');
END;
```

> **See Also:** "GET_SESSION_STATE Function" on page 1-73, "GET_NUMERIC_SESSION_STATE Function" on page 1-64, and "Understanding Session State Management" in *Oracle Application Express Application Builder User's Guide*

SET_SESSION_TERRITORY Procedure

This procedure sets the territory to be used for the current user in the current Application Express session. The territory name must be a valid Oracle territory.

Syntax

```
APEX_UTIL.SET_SESSION_TERRITORY(
    p_territory IN VARCHAR2);
```

Parameters

Table 1–83 describes the parameters available in the `SET_SESSION_TERRITORY` procedure.

Table 1–83 SET_SESSION_TERRITORYParameters

Parameter	Description
`p_territory`	A valid Oracle territory name. Examples include: AMERICA, UNITED KINGDOM, ISRAEL, AUSTRIA, and UNITED ARAB EMIRATES.

Example

The following example shows how to use the `SET_SESSION_TERRITORY` procedure. It sets the territory for the current user for the duration of the Application Express session.

```
BEGIN
    APEX_UTIL.SET_SESSION_TERRITORY( P_TERRITORY => 'UNITED KINGDOM');
END;
```

SET_SESSION_TIME_ZONE Procedure

This procedure sets the time zone to be used for the current user in the current Application Express session.

Syntax

```
APEX_UTIL.SET_SESSION_TIME_ZONE(
    p_time_zone IN VARCHAR2);
```

Parameters

Table 1–84 describes the parameters available in the SET_SESSION_TIME_ZONE procedure.

Table 1–84 SET_SESSION_TIME_ZONE Parameters

Parameter	Description
p_timezone	A time zone value in the form of hours and minutes. Examples include: +09:00, 04:00, -05:00.

Example

The following example shows how to use the SET_SESSION_TIME_ZONE procedure. It sets the time zone for the current user for the duration of the Application Express session.

```
BEGIN
    APEX_UTIL.SET_SESSION_TIME_ZONE( P_TIME_ZONE => '-05:00');
END;
```

SET_USERNAME Procedure

This procedure updates a user account with a new USER_NAME value. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
APEX_UTIL.SET_USERNAME(
    p_userid   IN NUMBER,
    p_username IN VARCHAR2);
```

Parameters

Table 1–85 describes the parameters available in the SET_USERNAME procedure.

Table 1–85 SET_USERNAME Parameters

Parameter	Description
p_userid	The numeric ID of the user account
p_username	USER_NAME value to be saved in the user account

Example

The following example shows how to use the SET_USERNAME procedure to set the value of USERNAME to 'USER-XRAY' for the user 'FRANK'.

```
BEGIN
    APEX_UTIL.SET_USERNAME(
        p_userid     => APEX_UTIL.GET_USER_ID('FRANK'),
        P_username   => 'USER-XRAY');
END;
```

See Also: "GET_USERNAME Function" on page 1-78 and "GET_USER_ID Function" on page 1-76

SHOW_SCREEN_READER_MODE_TOGGLE Procedure

This procedure displays a link to the current page to turn on or off, toggle, the mode. For example, if you are in standard mode, this function displays a link that when clicked switches the screen reader mode on.

Syntax

```
APEX_UTIL.SHOW_SCREEN_READER_MODE_TOGGLE (
    p_on_message  IN VARCHAR2 DEFAULT NULL,
    p_off_message IN VARCHAR2 DEFAULT NULL)
```

Parameters

Table 1–86 describes the parameters available in SHOW_SCREEN_READER_MODE_TOGGLE function.

Table 1–86 SHOW_SCREEN_READER_MODE_TOGGLE Parameters

Parameter	Description
p_on_message	Optional text used for the link to switch to screen reader mode, when you are in standard mode. If this parameter is not passed, the default 'Set Screen Reader Mode On' text will be displayed.
p_off_message	Optional text used for the link to switch to standard mode, when you are in screen reader mode. If this parameter is not passed, the default 'Set Screen Reader Mode Off' text will be displayed.

Example

When running in standard mode, this procedure will display a link 'Set Screen Reader Mode On', that when clicked refreshes the current page and switches on screen reader mode. When running in screen reader mode, a link 'Set Screen Reader Mode Off' is displayed, that when clicked refreshes the current page and switches back to standard mode.

```
BEGIN
    htp.p(apex_util.show_screen_reader_mode_toggle);
END;
```

STRING_TO_TABLE Function

Given a string, this function returns a PL/SQL array of type APEX_APPLICATION_GLOBAL.VC_ARR2. This array is a VARCHAR2(32767) table.

Syntax

```
APEX_UTIL.STRING_TO_TABLE (
    p_string       IN VARCHAR2,
    p_separator    IN VARCHAR2 DEFAULT ':')
    RETURN APEX_APPLICATION_GLOBAL.VC_ARR2;
```

Parameters

Table 1–87 describes the parameters available in the STRING_TO_TABLE function.

Table 1–87 STRING_TO_TABLE Parameters

Parameter	Description
p_string	String to be converted into a PL/SQL table of type APEX_APPLICATION_GLOBAL.VC_ARR2
p_separator	String separator. The default is a colon

Example

The following example shows how to use the STRING_TO_TABLE function. The function is passed the string 'One:Two:Three' in the p_string parameter and it returns a PL/SQL array of type APEX_APPLICATION_GLOBAL.VC_ARR2 containing 3 elements, the element at position 1 contains the value 'One', position 2 contains the value 'Two' and position 3 contains the value 'Three'. This is then output using the HTP.P function call.

```
DECLARE
    l_vc_arr2    APEX_APPLICATION_GLOBAL.VC_ARR2;
BEGIN
    l_vc_arr2 := APEX_UTIL.STRING_TO_TABLE('One:Two:Three');
    FOR z IN 1..l_vc_arr2.count LOOP
        htp.p(l_vc_arr2(z));
    END LOOP;
END;
```

See Also: "TABLE_TO_STRING Function" on page 1-133

STRONG_PASSWORD_CHECK Procedure

This procedure returns `Boolean OUT` values based on whether or not a proposed password meets the password strength requirements as defined by the Oracle Application Express site administrator.

Syntax

```
APEX_UTIL.STRONG_PASSWORD_CHECK(
    p_username                    IN  VARCHAR2,
    p_password                    IN  VARCHAR2,
    p_old_password                IN  VARCHAR2,
    p_workspace_name              IN  VARCHAR2,
    p_use_strong_rules            IN  BOOLEAN,
    p_min_length_err              OUT BOOLEAN,
    p_new_differs_by_err          OUT BOOLEAN,
    p_one_alpha_err               OUT BOOLEAN,
    p_one_numeric_err             OUT BOOLEAN,
    p_one_punctuation_err         OUT BOOLEAN,
    p_one_upper_err               OUT BOOLEAN,
    p_one_lower_err               OUT BOOLEAN,
    p_not_like_username_err       OUT BOOLEAN,
    p_not_like_workspace_name_err OUT BOOLEAN,
    p_not_like_words_err          OUT BOOLEAN,
    p_not_reusable_err            OUT BOOLEAN);
```

Parameters

Table 1–88 describes the parameters available in the `STRONG_PASSWORD_CHECK` procedure.

Table 1–88 STRONG_PASSWORD_CHECK Parameters

Parameter	Description
`p_username`	Username that identifies the account in the current workspace
`p_password`	Password to be checked against password strength rules
`p_old_password`	Current password for the account. Used only to enforce "new password must differ from old" rule
`p_workspace_name`	Current workspace name, used only to enforce "password must not contain workspace name" rule
`p_use_strong_rules`	Pass `FALSE` when calling this API
`p_min_length_err`	Result returns `True` or `False` depending upon whether the password meets minimum length requirement
`p_new_differs_by_err`	Result returns `True` or `False` depending upon whether the password meets "new password must differ from old" requirements
`p_one_alpha_err`	Result returns `True` or `False` depending upon whether the password meets requirement to contain at least one alphabetic character
`p_one_numeric_err`	Result returns `True` or `False` depending upon whether the password meets requirements to contain at least one numeric character

Table 1–88 (Cont.) STRONG_PASSWORD_CHECK Parameters

Parameter	Description
p_one_punctuation_err	Result returns True or False depending upon whether the password meets requirements to contain at least one punctuation character
p_one_upper_err	Result returns True or False depending upon whether the password meets requirements to contain at least one upper-case character
p_one_lower_err	Result returns True or False depending upon whether the password meets requirements to contain at least one lower-case character
p_not_like_username_err	Result returns True or False depending upon whether the password meets requirements that it not contain the username
p_not_like_workspace_name_err	Result returns True or False whether upon whether the password meets requirements that it not contain the workspace name
p_not_like_words_err	Result returns True or False whether the password meets requirements that it not contain specified simple words
p_not_reusable_err	Result returns True or False whether the password can be reused based on password history rules

Example

The following example shows how to use the STRONG_PASSWORD_CHECK procedure. It checks the new password 'foo' for the user 'SOMEBODY' meets all the password strength requirements defined by the Oracle Application Express site administrator. If any of the checks fail (the associated OUT parameter returns TRUE), then the example outputs a relevant message. For example, if the Oracle Application Express site administrator has defined that passwords must have at least one numeric character and the password 'foo' was checked, then the p_one_numeric_err OUT parameter would return TRUE and the message 'Password must contain at least one numeric character' would be output.

```
DECLARE
    l_username                  varchar2(30);
    l_password                  varchar2(30);
    l_old_password              varchar2(30);
    l_workspace_name            varchar2(30);
    l_min_length_err            boolean;
    l_new_differs_by_err        boolean;
    l_one_alpha_err             boolean;
    l_one_numeric_err           boolean;
    l_one_punctuation_err       boolean;
    l_one_upper_err             boolean;
    l_one_lower_err             boolean;
    l_not_like_username_err     boolean;
    l_not_like_workspace_name_err boolean;
    l_not_like_words_err        boolean;
    l_not_reusable_err          boolean;
    l_password_history_days     pls_integer;
BEGIN
    l_username := 'SOMEBODY';
    l_password := 'foo';
    l_old_password := 'foo';
    l_workspace_name := 'XYX_WS';
    l_password_history_days :=
        apex_instance_admin.get_parameter ('PASSWORD_HISTORY_DAYS');
```

```
APEX_UTIL.STRONG_PASSWORD_CHECK(
    p_username                    => l_username,
    p_password                    => l_password,
    p_old_password                => l_old_password,
    p_workspace_name              => l_workspace_name,
    p_use_strong_rules            => false,
    p_min_length_err              => l_min_length_err,
    p_new_differs_by_err          => l_new_differs_by_err,
    p_one_alpha_err               => l_one_alpha_err,
    p_one_numeric_err             => l_one_numeric_err,
    p_one_punctuation_err         => l_one_punctuation_err,
    p_one_upper_err               => l_one_upper_err,
    p_one_lower_err               => l_one_lower_err,
    p_not_like_username_err       => l_not_like_username_err,
    p_not_like_workspace_name_err => l_not_like_workspace_name_err,
    p_not_like_words_err          => l_not_like_words_err,
    p_not_reusable_err            => l_not_reusable_err);

IF l_min_length_err THEN
    htp.p('Password is too short');
END IF;

IF l_new_differs_by_err THEN
    htp.p('Password is too similar to the old password');
END IF;

IF l_one_alpha_err THEN
    htp.p('Password must contain at least one alphabetic character');
END IF;

IF l_one_numeric_err THEN
    htp.p('Password  must contain at least one numeric character');
END IF;

IF l_one_punctuation_err THEN
    htp.p('Password  must contain at least one punctuation character');
END IF;

IF l_one_upper_err THEN
    htp.p('Password must contain at least one upper-case character');
END IF;

IF l_one_lower_err THEN
    htp.p('Password must contain at least one lower-case character');
END IF;

IF l_not_like_username_err THEN
    htp.p('Password may not contain the username');
END IF;

IF l_not_like_workspace_name_err THEN
    htp.p('Password may not contain the workspace name');
END IF;

IF l_not_like_words_err THEN
    htp.p('Password contains one or more prohibited common words');
END IF;

IF l_not_reusable_err THEN
```

```
        htp.p('Password cannot be used because it has been used for the account
within the last '||l_password_history_days||' days.');
    END IF;
END;
```

See Also: "About Password Policies" in *Oracle Application Express Administration Guide*

STRONG_PASSWORD_VALIDATION Function

This function returns formatted HTML in a VARCHAR2 result based on whether or not a proposed password meets the password strength requirements as defined by the Oracle Application Express site administrator.

Syntax

```
FUNCTION STRONG_PASSWORD_VALIDATION(
    p_username                     IN  VARCHAR2,
    p_password                     IN  VARCHAR2,
    P_OLD_PASSWORD                 IN  VARCHAR2 DEFAULT NULL,
    P_WORKSPACE_NAME               IN  VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 1–89 describes the parameters available in the STRONG_PASSWORD_VALIDATION function.

Table 1–89 STRONG_PASSWORD_VALIDATION Parameters

Parameter	Description
p_username	Username that identifies the account in the current workspace
p_password	Password to be checked against password strength rules
p_old_password	Current password for the account. Used only to enforce "new password must differ from old" rule
p_workspace_name	Current workspace name, used only to enforce "password must not contain workspace name" rule

Example

The following example shows how to use the STRONG_PASSWORD_VALIDATION procedure. It checks the new password 'foo' for the user 'SOMEBODY' meets all the password strength requirements defined by the Oracle Application Express site administrator. If any of the checks fail, then the example outputs formatted HTML showing details of where the new password fails to meet requirements.

```
DECLARE
      l_username                   varchar2(30);
      l_password                   varchar2(30);
      l_old_password               varchar2(30);
      l_workspace_name             varchar2(30);
BEGIN
    l_username := 'SOMEBODY';
    l_password := 'foo';
    l_old_password := 'foo';
    l_workspace_name := 'XYX_WS';

    HTP.P(APEX_UTIL.STRONG_PASSWORD_VALIDATION(
        p_username                   => l_username,
        p_password                   => l_password,
        p_old_password               => l_old_password,
        p_workspace_name             => l_workspace_name));
END;
```

SUBMIT_FEEDBACK Procedure

This procedure allows you to write a procedure to submit feedback, rather than using the page that can be generated by create page of type feedback.

Syntax

```
APEX_UTIL.SUBMIT_FEEDBACK (
    p_comment          IN VARCHAR2 DEFAULT NULL,
    p_type             IN VARCHAR2 DEFAULT '1',
    p_application_id   IN VARCHAR2 DEFAULT NULL,
    p_page_id          IN VARCHAR2 DEFAULT NULL,
    p_email            IN VARCHAR2 DEFAULT NULL,
    p_screen_width     IN VARCHAR2 DEFAULT NULL,
    p_screen_height    IN VARCHAR2 DEFAULT NULL,
    p_attribute_01     IN VARCHAR2 DEFAULT NULL,
    p_attribute_02     IN VARCHAR2 DEFAULT NULL,
    p_attribute_03     IN VARCHAR2 DEFAULT NULL,
    p_attribute_04     IN VARCHAR2 DEFAULT NULL,
    p_attribute_05     IN VARCHAR2 DEFAULT NULL,
    p_attribute_06     IN VARCHAR2 DEFAULT NULL,
    p_attribute_07     IN VARCHAR2 DEFAULT NULL,
    p_attribute_08     IN VARCHAR2 DEFAULT NULL,
    p_label_01         IN VARCHAR2 DEFAULT NULL,
    p_label_02         IN VARCHAR2 DEFAULT NULLl,
    p_label_03         IN VARCHAR2 DEFAULT NULL,
    p_label_04         IN VARCHAR2 DEFAULT NULL,
    p_label_05         IN VARCHAR2 DEFAULT NULL,
    p_label_06         IN VARCHAR2 DEFAULT NULL,
    p_label_07         IN VARCHAR2 DEFAULT NULL,
    p_label_08         IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 1–90 describes the parameters available in the `SUBMIT_FEEDBACK` procedure.

Table 1–90 SUBMIT_FEEDBACK Parameters

Parameter	Description
`p_comment`	comment to be submitted
`p_type`	type of feedback (1 is General Comment, 2 is Enhancement Request, 3 is Bug)
`p_application_id`	id of application related to the feedback
`p_page_id`	id of page related to the feedback
`p_email`	email of the user providing the feedback
`p_screen_width`	width of screen at time feedback was provided
`p_screen_height`	height of screen at time feedback was provided
`p_attribute_01`	Custom attribute for collecting feedback
`p_attribute_02`	Custom attribute for collecting feedback
`p_attribute_03`	Custom attribute for collecting feedback
`p_attribute_04`	Custom attribute for collecting feedback
`p_attribute_05`	Custom attribute for collecting feedback

Table 1–90 (Cont.) SUBMIT_FEEDBACK Parameters

Parameter	Description
p_attribute_06	Custom attribute for collecting feedback
p_attribute_07	Custom attribute for collecting feedback
p_attribute_08	Custom attribute for collecting feedback
p_label_01	Label for corresponding custom attribute
p_label_02	Label for corresponding custom attribute
p_label_03	Label for corresponding custom attribute
p_label_04	Label for corresponding custom attribute
p_label_05	Label for corresponding custom attribute
p_label_06	Label for corresponding custom attribute
p_label_07	Label for corresponding custom attribute
p_label_08	Label for corresponding custom attribute

SUBMIT_FEEDBACK_FOLLOWUP Procedure

This procedure allows you to submit followup to a feedback.

Syntax

```
APEX_UTIL.SUBMIT_FEEDBACK_FOLLOWUP (
    p_feedback_id      IN NUMBER,
    p_follow_up        IN VARCHAR2 DEFAULT NULL,
    p_email            IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 1–91 describes the parameters available in the `SUBMIT_FEEDBACK_FOLLOWUP` procedure.

Table 1–91 SUBMIT_FEEDBACK_FOLLOWUP Parameters

Parameter	Description
`p_feedback_followup`	id of feedback that this is a followup to
`p_follow_up`	text of followup
`p_email`	email of user providing the followup

TABLE_TO_STRING Function

Given a a PL/SQL table of type APEX_APPLICATION_GLOBAL.VC_ARR2, this function returns a delimited string separated by the supplied separator, or by the default separator, a colon (:).

Syntax

```
APEX_UTIL.TABLE_TO_STRING (
    p_table       IN     APEX_APPLICATION_GLOBAL.VC_ARR2,
    p_string      IN     VARCHAR2 DEFAULT ':')
RETURN VARCHAR2;
```

Parameters

Table 1–92 describes the parameters available in the TABLE_TO_STRING function.

Table 1–92 TABLE_TO_STRING Parameters

Parameter	Description
p_string	String separator. Default separator is a colon (:)
p_table	PL/SQL table that is to be converted into a delimited string

Example

The following example shows how to use the TABLE_TO_STRING function. The example first calls STRING_TO_TABLE which is passed the string 'One:Two:Three' in the p_string parameter, and returns a PL/SQL array of type APEX_APPLICATION_GLOBAL.VC_ARR2 containing 3 elements, the element at position 1 contains the value 'One', position 2 contains the value 'Two' and position 3 contains the value 'Three'. This array is then passed in to the TABLE_TO_STRING function in the p_string parameter, which then returns back the original string 'One:Two:Three'.

```
DECLARE
    l_string     VARCHAR2(255);
    l_vc_arr2    APEX_APPLICATION_GLOBAL.VC_ARR2;
BEGIN
    l_vc_arr2 := APEX_UTIL.STRING_TO_TABLE('One:Two:Three');
    l_string := APEX_UTIL.TABLE_TO_STRING(l_vc_arr2);
END;
```

See Also: "STRING_TO_TABLE Function" on page 1-124

UNEXPIRE_END_USER_ACCOUNT Procedure

Makes expired end users accounts and the associated passwords usable, enabling a end user to log in to developed applications.

Syntax

```
APEX_UTIL.UNEXPIRE_END_USER_ACCOUNT (
    p_user_name IN VARCHAR2);
```

Parameters

Table 1–93 describes the parameters available in the UNEXPIRE_END_USER_ACCOUNT procedure.

Table 1–93 UNEXPIRE_END_USER_ACCOUNT Parameters

Parameter	Description
p_user_name	The user name of the user account

Example

The following example shows how to use the UNEXPIRE_END_USER_ACCOUNT procedure. Use this procedure to renew (unexpire) an Application Express end user account in the current workspace. This action specifically renews the account for use by end users to authenticate to developed applications and may also renew the account for use by developers or administrators to log in to a workspace.

This procedure must be run by a user having administration privileges in the current workspace.

```
BEGIN
    FOR c1 IN (SELECT user_name from wwv_flow_users) LOOP
        APEX_UTIL.UNEXPIRE_END_USER_ACCOUNT(p_user_name => c1.user_name);
        htp.p('End User Account:'||c1.user_name||' is now valid.');
    END LOOP;
END;
```

See Also: " EXPIRE_END_USER_ACCOUNT Parameters" on page 1-32 and "END_USER_ACCOUNT_DAYS_LEFT Function" on page 1-31

UNEXPIRE_WORKSPACE_ACCOUNT Procedure

Unexpires developer and workspace administrator accounts and the associated passwords, enabling the developer or administrator to log in to a workspace.

Syntax

```
APEX_UTIL.UNEXPIRE_WORKSPACE_ACCOUNT (
    p_user_name IN VARCHAR2);
```

Parameters

Table 1–94 describes the parameters available in the UNEXPIRE_WORKSPACE_ACCOUNT procedure.

Table 1–94 UNEXPIRE_WORKSPACE_ACCOUNT Parameters

Parameter	Description
p_user_name	The user name of the user account

Example

The following example shows how to use the UNEXPIRE_WORKSPACE_ACCOUNT procedure. Use this procedure to renew (unexpire) an Application Express workspace administrator account in the current workspace. This action specifically renews the account for use by developers or administrators to login to a workspace and may also renew the account with respect to its use by end users to authenticate to developed applications.

This procedure must be run by a user having administration privileges in the current workspace.

```
BEGIN
    FOR c1 IN (select user_name from wwv_flow_users) loop
        APEX_UTIL.UNEXPIRE_WORKSPACE_ACCOUNT(p_user_name => c1.user_name);
        htp.p('Workspace Account:'||c1.user_name||' is now valid.');
    END LOOP;
END;
```

See Also: "EXPIRE_WORKSPACE_ACCOUNT Procedure" on page 1-33 and "WORKSPACE_ACCOUNT_DAYS_LEFT Function" on page 1-139

UNLOCK_ACCOUNT Procedure

Sets a user account status to unlocked. Must be run by an authenticated workspace administrator in a page request context.

Syntax

```
APEX_UTIL.UNLOCK_ACCOUNT (
     p_user_name IN VARCHAR2);
```

Parameters

Table 1–95 describes the parameters available in the UNLOCK_ACCOUNT procedure.

Table 1–95 UNLOCK_ACCOUNT Parameters

Parameter	Description
p_user_name	The user name of the user account

Example

The following example shows how to use the UNLOCK_ACCOUNT procedure. Use this procedure to unlock an Application Express account in the current workspace. This action unlocks the account for use by administrators, developers, and end users.

This procedure must be run by a user who has administration privileges in the current workspace

```
BEGIN
    FOR c1 IN (SELECT user_name from wwv_flow_users) LOOP
        APEX_UTIL.UNLOCK_ACCOUNT(p_user_name => c1.user_name);
        htp.p('End User Account:'||c1.user_name||' is now unlocked.');
    END LOOP;
END;
```

See Also: "LOCK_ACCOUNT Procedure" on page 1-88 and "GET_ACCOUNT_LOCKED_STATUS Function" on page 1-46

URL_ENCODE Function

The following special characters are encoded as follows:

Special Characters	After Encoding
%	%25
+	%2B
space	+
.	%2E
*	%2A
?	%3F
\	%5C
/	%2F
>	%3E
<	%3C
}	%7B
{	%7D
~	%7E
[	%5B
]	%5D
’	%60
;	%3B
?	%3F
@	%40
&	%26
#	%23
|	%7C
^	%5E
:	%3A
=	%3D
$	%24

Syntax

```
APEX_UTIL.URL_ENCODE (
    p_url   IN    VARCHAR2)
    RETURN VARCHAR2;
```

Parameters

Table 1–96 describes the parameters available in the URL_ENCODE function.

Table 1–96 URL_ENCODE Parameters

Parameter	Description
p_url	The string to be encoded

Example

The following example shows how to use the URL_ENCODE function.

```
DECLARE
    l_url  VARCHAR2(255);
BEGIN
    l_url := APEX_UTIL.URL_ENCODE('http://www.myurl.com?id=1&cat=foo');
END;
```

In this example, the following URL:

```
http://www.myurl.com?id=1&cat=foo
```

Would be returned as:

```
http%3A%2F%2Fwww%2Emyurl%2Ecom%3Fid%3D1%26cat%3Dfoo
```

WORKSPACE_ACCOUNT_DAYS_LEFT Function

Returns the number of days remaining before the developer or workspace administrator account password expires. This function may be run in a page request context by any authenticated user.

Syntax

```
APEX_UTIL.WORKSPACE_ACCOUNT_DAYS_LEFT (
    p_user_name IN VARCHAR2)
    RETURN NUMBER;
```

Parameters

Table 1–97 describes the parameters available in the `WORKSPACE_ACCOUNT_DAYS_LEFT` procedure.

Table 1–97 WORKSPACE_ACCOUNT_DAYS_LEFT Parameters

Parameter	Description
`p_user_name`	The user name of the user account

Example

The following example shows how to use the `WORKSPACE_ACCOUNT_DAYS_LEFT` function. It can be used in to find the number of days remaining before an Application Express administrator or developer account in the current workspace expires.

```
DECLARE
    l_days_left NUMBER;
BEGIN
    FOR c1 IN (SELECT user_name from wwv_flow_users) LOOP
        l_days_left := APEX_UTIL.WORKSPACE_ACCOUNT_DAYS_LEFT(p_user_name =>
c1.user_name)
        htp.p('Workspace Account:'||c1.user_name||' will expire in '||l_days_
left||' days.');
    END LOOP;
END;
```

See Also: "EXPIRE_WORKSPACE_ACCOUNT Procedure" on page 1-33 and "UNEXPIRE_WORKSPACE_ACCOUNT Procedure" on page 1-135

2

APEX_MAIL

You can use the APEX_MAIL package to send an email from an Oracle Application Express application. This package is built on top of the Oracle supplied UTL_SMTP package. Because of this dependence, the UTL_SMTP package must be installed and functioning in order to use APEX_MAIL.

> **See Also:** *Oracle Database PL/SQL Packages and Types Reference* for more information about the UTL_SMTP package

APEX_MAIL contains three procedures. Use APEX_MAIL.SEND to send an outbound email message from your application. Use APEX_MAIL.PUSH_QUEUE to deliver mail messages stored in APEX_MAIL_QUEUE. Use APEX_MAIL.ADD_ATTACHMENT to send an outbound email message from your application as an attachment.

Topics:

- About Configuring Oracle Application Express to Send Email
- ADD_ATTACHMENT Procedure
- PUSH_QUEUE Procedure
- SEND Procedure

Note: The most efficient approach to sending email is to create a background job (using the DBMS_JOB or DBMS_SCHEDULER package) to periodically send all mail messages stored in the active mail queue. In order to call the APEX_MAIL package from outside the context of an Application Express application, you must call apex_util.set_security_group_id as in the following example:

```
for c1 in (
   select workspace_id
     from apex_applications
    where application_id = p_app_id )
loop
   apex_util.set_security_group_id(p_security_group_id =>
c1.workspace_id);
end loop;
```

> **See Also:** "Sending Email from an Application" in *Oracle Application Express Application Builder User's Guide*

About Configuring Oracle Application Express to Send Email

Before you can send email from an Application Builder application, you must:

1. Log in to Oracle Application Express Administration Services and configure the email settings on the Instance Settings page. See in Oracle Application Express Administration Guide.
2. If you are running Oracle Application Express with Oracle Database 11*g* release 1 (11.1), you must enable outbound mail. In Oracle Database 11*g* release 1 (11.1), the ability to interact with network services is disabled by default. See "Enabling Network Services in Oracle Database 11g" in *Oracle Application Express Application Builder User's Guide*.

   Tip: You can configure Oracle Application Express to automatically email users their login credentials when a new workspace request has been approved. To learn more, see "Specifying a Provisioning Mode"in *Oracle Application Express Administration Guide*.

ADD_ATTACHMENT Procedure

This procedure sends an outbound email message from an application as an attachment. To add multiple attachments to a single email, APEX_MAIL.ADD_ATTACHMENT can be called repeatedly for a single email message.

Syntax

```
APEX_MAIL.ADD_ATTACHMENT(
    p_mail_id                   IN    NUMBER,
    p_attachment                IN    BLOB,
    p_filename                  IN    VARCHAR2,
    p_mime_type                 IN    VARCHAR2);
```

Parameters

Table 2–1 describes the parameters available in the ADD_ATTACHMENT procedure.

Table 2–1 ADD_ATTACHMENT Parameters

Parameter	Description
p_mail_id	The numeric ID associated with the email. This is the numeric identifier returned from the call to APEX_MAIL.SEND to compose the email body.
p_attachment	A BLOB variable containing the binary content to be attached to the email message.
p_filename	The filename associated with the email attachment.
p_mime_type	A valid MIME type (or Internet media type) to associate with the email attachment.

Examples

The following example demonstrates how to access files stored in APEX_APPLICATION_FILES and add them to an outbound email message

```
DECLARE
    l_id NUMBER;
BEGIN
    l_id := APEX_MAIL.SEND(
        p_to        => 'fred@flintstone.com',
        p_from      => 'barney@rubble.com',
        p_subj      => 'APEX_MAIL with attachment',
        p_body      => 'Please review the attachment.',
        p_body_html => '<b>Please</b> review the attachment');
    FOR c1 IN (SELECT filename, blob_content, mime_type
        FROM APEX_APPLICATION_FILES
        WHERE ID IN (123,456)) LOOP

        APEX_MAIL.ADD_ATTACHMENT(
            p_mail_id    => l_id,
            p_attachment => c1.blob_content,
            p_filename   => c1.filename,
            p_mime_type  => c1.mime_type);
        END LOOP;
    COMMIT;
END;
/
```

PUSH_QUEUE Procedure

Oracle Application Express stores unsent email messages in a table named APEX_MAIL_QUEUE. You can manually deliver mail messages stored in this queue to the specified SMTP gateway by invoking the APEX_MAIL.PUSH_QUEUE procedure.

Oracle Application Express logs successfully submitted message in the table APEX_MAIL_LOG with the timestamp reflecting your server's local time. Keep in mind, the most efficient approach to sending email is to create a background job (using a DBMS_JOB package) to periodically send all mail messages stored in the active mail queue.

> **See Also:** "Sending an Email from an Application" in *Oracle Application Express Application Builder User's Guide*

Syntax

```
APEX_MAIL.PUSH_QUEUE(
    p_smtp_hostname             IN    VARCHAR2 DEFAULT NULL,
    p_smtp_portno               IN    NUMBER   DEFAULT NULL);
```

Parameters

Table 2–2 describes the parameters available in the PUSH_QUEUE procedure.

Table 2–2 PUSH_QUEUE Parameters

Parameters	Description
p_smtp_hostname	SMTP gateway host name
p_smtp_portno	SMTP gateway port number

Note that these parameter values are provided for backward compatibility, but their respective values are ignored. The SMTP gateway hostname and SMTP gateway port number are exclusively derived from values entered on the Manage Environment Settings when sending email.

> **See Also:** "Configuring Email Settings" in *Oracle Application Express Administration Guide*

Example

The following example demonstrates the use of the APEX_MAIL.PUSH_QUEUE procedure using a shell script. This example only applies to UNIX/LINUX installations.

```
SQLPLUS / <<EOF
APEX_MAIL.PUSH_QUEUE;
DISCONNECT
EXIT
EOF
```

> **See Also:** "Sending Email from an Application" in *Oracle Application Express Application Builder User's Guide*

SEND Procedure

This procedure sends an outbound email message from an application. Although you can use this procedure to pass in either a VARCHAR2 or a CLOB to p_body and p_body_html, the data types must be the same. In other words, you cannot pass a CLOB to P_BODY and a VARCHAR2 to p_body_html.

When using APEX_MAIL.SEND, remember the following:

- **No single line may exceed 1000 characters.** The SMTP/MIME specification dictates that no single line shall exceed 1000 characters. To comply with this restriction, you must add a carriage return or line feed characters to break up your p_body or p_body_html parameters into chunks of 1000 characters or less. Failing to do so will result in erroneous email messages, including partial messages or messages with extraneous exclamation points.
- **Plain text and HTML email content.** Passing a value to p_body, but not p_body_html results in a plain text message. Passing a value to p_body and p_body_html yields a multi-part message that includes both plain text and HTML content. The settings and capabilities of the recipient's email client determine what displays. Although most modern email clients can read an HTML formatted email, remember that some users disable this functionality to address security issues.
- **Avoid images.** When referencing images in p_body_html using the <img /> tag, remember that the images must be accessible to the recipient's email client in order for them to see the image.

  For example, suppose you reference an image on your network called hello.gif as follows:

  ```
  <img src="http://someserver.com/hello.gif" alt="Hello" />]
  ```

  In this example, the image is not attached to the email, but is referenced by the email. For the recipient to see it, they must be able to access the image using a Web browser. If the image is inside a firewall and the recipient is outside of the firewall, the image will not display. For this reason, avoid using images. If you must include images, be sure to include the ALT attribute to provide a textual description in the event the image is not accessible.

Syntax

```
APEX_MAIL.SEND(
    p_to                        IN    VARCHAR2,
    p_from                      IN    VARCHAR2,
    p_body                      IN  [ VARCHAR2 | CLOB ],
    p_body_html                 IN  [ VARCHAR2 | CLOB ] DEFAULT NULL,
    p_subj                      IN    VARCHAR2 DEFAULT NULL,
    p_cc                        IN    VARCHAR2 DEFAULT NULL,
    p_bcc                       IN    VARCHAR2 DEFAULT NULL,
    p_replyto                   IN    VARCHAR2);
```

Parameters

Table 2–3 describes the parameters available in the SEND procedure.

Table 2–3 SEND Parameters

Parameter	Description
p_to	Valid email address to which the email will be sent (required). For multiple email addresses, use a comma-separated list
p_from	Email address from which the email will be sent (required). This email address must be a valid address. Otherwise, the message will not be sent
p_body	Body of the email in plain text, not HTML (required). If a value is passed to p_body_html, then this is the only text the recipient sees. If a value is not passed to p_body_html, then this text only displays for email clients that do not support HTML or have HTML disabled. A carriage return or line feed (CRLF) must be included every 1000 characters.
p_body_html	Body of the email in HTML format. This must be a full HTML document including the <html> and <body> tags. A single line cannot exceed 1000 characters without a carriage return or line feed (CRLF)
p_subj	Subject of the email
p_cc	Valid email addresses to which the email is copied. For multiple email addresses, use a comma-separated list
p_bcc	Valid email addresses to which the email is blind copied. For multiple email addresses, use a comma-separated list
p_replyto	Address of the Reply-To mail header. You can use this parameter as follows: ■ If you omit the p_replyto parameter, the Reply-To mail header is set to the value specified in the p_from parameter ■ If you include the p_replyto parameter, but provide a NULL value, the Reply-To mail header is set to NULL. This results in the suppression of automatic email replies ■ If you include p_replyto parameter, but provide a non-null value (for example, a valid email address), you will send these messages, but the automatic replies will go to the value specified (for example, the email address)

Examples

The following example demonstrates how to use APEX_MAIL.SEND to send a plain text email message from an application.

```
-- Example One: Plain Text only message
DECLARE
    l_body      CLOB;
BEGIN
    l_body := 'Thank you for your interest in the APEX_MAIL
package.'||utl_tcp.crlf||utl_tcp.crlf;
    l_body := l_body ||'  Sincerely,'||utl_tcp.crlf;
    l_body := l_body ||'  The APEX Dev Team'||utl_tcp.crlf;
    apex_mail.send(
        p_to       => 'some_user@somewhere.com',   -- change to your email address
        p_from     => 'some_sender@somewhere.com', -- change to a real senders
email address
        p_body     => l_body,
        p_subj     => 'APEX_MAIL Package - Plain Text message');
END;
/
```

The following example demonstrates how to use `APEX_MAIL.SEND` to send an HTML email message from an application. Remember, you must include a carriage return or line feed (CRLF) every 1000 characters. The example that follows uses `utl_tcp.crlf`.

```
-- Example Two: Plain Text / HTML message
DECLARE
    l_body      CLOB;
    l_body_html CLOB;
BEGIN
    l_body := 'To view the content of this message, please use an HTML enabled
mail client.'||utl_tcp.crlf;

    l_body_html := '<html>
        <head>
            <style type="text/css">
                body{font-family: Arial, Helvetica, sans-serif;
                    font-size:10pt;
                    margin:30px;
                    background-color:#ffffff;}

                span.sig{font-style:italic;
                    font-weight:bold;
                    color:#811919;}
             </style>
         </head>
         <body>'||utl_tcp.crlf;
    l_body_html := l_body_html ||'<p>Thank you for your interest in the
<strong>APEX_MAIL</strong> package.</p>'||utl_tcp.crlf;
    l_body_html := l_body_html ||'  Sincerely,<br />'||utl_tcp.crlf;
    l_body_html := l_body_html ||'  <span class="sig">The APEX Dev Team</span><br
/>'||utl_tcp.crlf;
    apex_mail.send(
    p_to   => 'some_user@somewhere.com',   -- change to your email address
    p_from => 'some_sender@somewhere.com', -- change to a real senders email
address
    p_body      => l_body,
    p_body_html => l_body_html,
    p_subj      => 'APEX_MAIL Package - HTML formatted message');
END;
/
```

3

APEX_ITEM

You can use the `APEX_ITEM` package to create form elements dynamically based on a SQL query instead of creating individual items page by page.

Topics:

- CHECKBOX Function
- DATE_POPUP Function
- DISPLAY_AND_SAVE Function
- HIDDEN Function
- MD5_CHECKSUM Function
- MD5_HIDDEN Function
- POPUP_FROM_LOV Function
- POPUP_FROM_QUERY Function
- POPUPKEY_FROM_LOV Function
- POPUPKEY_FROM_QUERY Function
- RADIOGROUP Function
- SELECT_LIST Function
- SELECT_LIST_FROM_LOV Function
- SELECT_LIST_FROM_LOV_XL Function
- SELECT_LIST_FROM_QUERY Function
- SELECT_LIST_FROM_QUERY_XL Function
- TEXT Function
- TEXTAREA Function
- TEXT_FROM_LOV Function
- TEXT_FROM_LOV_QUERY Function

CHECKBOX Function

This function creates check boxes.

Syntax

```
APEX_ITEM.CHECKBOX(
    p_idx                       IN    NUMBER,
    p_value                     IN    VARCHAR2 DEFAULT NULL,
    p_attributes                IN    VARCHAR2 DEFAULT NULL,
    p_checked_values            IN    VARCHAR2 DEFAULT NULL,
    p_checked_values_delimitor  IN    VARCHAR2 DEFAULT ':',
    p_item_id                   IN    VARCHAR2 DEFAULT NULL,
    p_item_label                IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 3–1 describes the parameters available in the CHECKBOX function.

Table 3–1 CHECKBOX Parameters

Parameter	Description
p_idx	Number that determines which APEX_APPLICATION global variable will be used. Valid range of values is 1 to 50. For example 1 creates F01 and 2 creates F02
p_value	Value of a check box, hidden field, or input form item
p_attributes	Controls HTML tag attributes (such as disabled)
p_checked_values	Values to be checked by default
p_checked_values_delimitor	Delimits the values in the previous parameter, p_checked_values
p_item_id	HTML attribute ID for the <input> tag
p_item_label	Invisible label created for the item

Examples of Default Check Box Behavior

The following example demonstrates how to create a selected check box for each employee in the emp table.

```
SELECT APEX_ITEM.CHECKBOX(1,empno,'CHECKED') "Select",
    ename, job
FROM   emp
ORDER BY 1
```

The following example demonstrates how to have all check boxes for employees display without being selected.

```
SELECT APEX_ITEM.CHECKBOX(1,empno) "Select",
    ename, job
FROM   emp
ORDER BY 1
```

The following example demonstrates how to select the check boxes for employees who work in department 10.

```
SELECT APEX_ITEM.CHECKBOX(1,empno,DECODE(deptno,10,'CHECKED',NULL)) "Select",
    ename, job
FROM   emp
ORDER BY 1
```

The next example demonstrates how to select the check boxes for employees who work in department 10 or department 20.

```
SELECT APEX_ITEM.CHECKBOX(1,deptno,NULL,'10:20',':') "Select",
    ename, job
FROM   emp
ORDER BY 1
```

Creating an On-Submit Process

If you are using check boxes in your application, you might need to create an On Submit process to perform a specific type of action on the selected rows. For example, you could have a Delete button that utilizes the following logic:

```
SELECT APEX_ITEM.CHECKBOX(1,empno) "Select",
    ename, job
FROM   emp
ORDER  by 1
```

Consider the following sample on-submit process:

```
FOR I in 1..APEX_APPLICATION.G_F01.COUNT LOOP
    DELETE FROM emp WHERE empno = to_number(APEX_APPLICATION.G_F01(i));
END LOOP;
```

The following example demonstrates how to create unselected checkboxes for each employee in the emp table, with a unique ID. This is useful for referencing records from within JavaScript code:

```
SELECT APEX_ITEM.CHECKBOX(1,empno,NULL,NULL,NULL,'f01_#ROWNUM#') "Select",
    ename, job
FROM   emp
ORDER BY 1
```

DATE_POPUP Function

Use this function with forms that include date fields. The DATE_POPUP function dynamically generates a date field that has a popup calendar button.

Syntax

```
APEX_ITEM.DATE_POPUP(
    p_idx                     IN    NUMBER,
    p_row                     IN    NUMBER,
    p_value                   IN    VARCHAR2 DEFAULT NULL,
    p_date_format             IN    DATE DEFAULT 'DD-MON-YYYY',
    p_size                    IN    NUMBER DEFAULT 20,
    p_maxlength               IN    NUMBER DEFAULT 2000,
    p_attributes              IN    VARCHAR2 DEFAULT NULL,
    p_item_id                 IN    VARCHAR2 DEFAULT NULL,
    p_item_label              IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 3–2 describes the parameters available in the DATE_POPUP function.

Table 3–2 DATE_POPUP Parameters

Parameter	Description
p_idx	Number that determines which APEX_APPLICATION global variable will be used.Valid range of values is 1 to 50. For example, 1 creates F01 and 2 creates F02
p_row	This parameter is deprecated. Anything specified for this value will be ignored
p_value	Value of a field item
p_date_format	Valid database date format
p_size	Controls HTML tag attributes (such as disabled)
p_maxlength	Determines the maximum number of enterable characters. Becomes the maxlength attribute of the <input> HTML tag
p_attributes	Extra HTML parameters you want to add
p_item_id	HTML attribute ID for the <input> tag
p_item_label	Invisible label created for the item

> **See Also:** *Oracle Database SQL Language Reference* for information about the TO_CHAR or TO_DATE functions

Example

The following example demonstrates how to use APEX_ITEM.DATE_POPUP to create popup calendar buttons for the hiredate column.

```
SELECT
    empno,
    APEX_ITEM.HIDDEN(1,empno)||
    APEX_ITEM.TEXT(2,ename) ename,
```

```
    APEX_ITEM.TEXT(3,job) job,
    mgr,
    APEX_ITEM.DATE_POPUP(4,rownum,hiredate,'dd-mon-yyyy') hd,
    APEX_ITEM.TEXT(5,sal) sal,
    APEX_ITEM.TEXT(6,comm) comm,
    deptno
FROM emp
ORDER BY 1
```

DISPLAY_AND_SAVE Function

Use this function to display an item as text, but save its value to session state.

Syntax

```
APEX_ITEM.DISPLAY_AND_SAVE(
    p_idx         IN    NUMBER,
    p_value       IN    VARCHAR2 DEFAULT NULL,
    p_item_id     IN    VARCHAR2 DEFAULT NULL,
    p_item_label  IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 3–3 describes the parameters available in the `DISPLAY_AND_SAVE` function.

Table 3–3 DISPLAY_AND_SAVE Parameters

Parameter	Description
`p_idx`	Number that determines which `APEX_APPLICATION` global variable will be used.Valid range of values is 1 to 50. For example, 1 creates `F01` and 2 creates `F02`
`p_value`	Current value
`p_item_id`	HTML attribute ID for the `<span>` tag
`p_item_label`	Invisible label created for the item

Example

The following example demonstrates how to use the `APEX_ITEM.DISPLAY_AND_SAVE` function.

```
SELECT APEX_ITEM.DISPLAY_AND_SAVE(10,empno) c FROM emp
```

HIDDEN Function

This function dynamically generates hidden form items.

Syntax

```
APEX_ITEM.HIDDEN(
    p_idx         IN    NUMBER,
    p_value       IN    VARCHAR2 DEFAULT
    p_attributes  IN    VARCHAR2 DEFAULT NULL,
    p_item_id     IN    VARCHAR2 DEFAULT NULL,
    p_item_label  IN    VARCHAR2 DEFAULT NULL
) RETURN VARCHAR2;
```

Parameters

Table 3–4 describes the parameters available in the `HIDDEN` function.

Table 3–4 HIDDEN Parameters

Parameter	Description
`p_idx`	Number to identify the item you want to generate. The number will determine which `G_FXX` global is populated **See Also:** "APEX_APPLICATION" on page 4-1
`p_value`	Value of the hidden input form item
`p_attributes`	Extra HTML parameters you want to add
`p_item_id`	HTML attribute ID for the `<input>` tag
`p_item_label`	Invisible label created for the item

Example

Typically, the primary key of a table is stored as a hidden column and used for subsequent update processing, for example:

```
SELECT
    empno,
    APEX_ITEM.HIDDEN(1,empno)||
    APEX_ITEM.TEXT(2,ename) ename,
    APEX_ITEM.TEXT(3,job) job,
    mgr,
    APEX_ITEM.DATE_POPUP(4,rownum,hiredate,'dd-mon-yyyy') hiredate,
    APEX_ITEM.TEXT(5,sal) sal,
    APEX_ITEM.TEXT(6,comm) comm,
    deptno
FROM emp
ORDER BY 1
```

The previous query could use the following page process to process the results:

```
BEGIN
    FOR i IN 1..APEX_APPLICATION.G_F01.COUNT LOOP
        UPDATE emp
            SET
                ename=APEX_APPLICATION.G_F02(i),
                job=APEX_APPLICATION.G_F03(i),
```

```
                    hiredate=to_date(APEX_APPLICATION.G_F04(i),'dd-mon-yyyy'),
                    sal=APEX_APPLICATION.G_F05(i),
                    comm=APEX_APPLICATION.G_F06(i)
        WHERE empno=to_number(APEX_APPLICATION.G_F01(i));
    END LOOP;
END;
```

Note that the `G_F01` column (which corresponds to the hidden `EMPNO`) is used as the key to update each row.

MD5_CHECKSUM Function

This function is used for lost update detection. Lost update detection ensures data integrity in applications where data can be accessed concurrently.

This function produces hidden form field(s) with a name attribute equal to 'fcs' and includes 50 inputs. APEX_ITEM.MD5_CHECKSUM also produces an MD5 checksum using the Oracle database DBMS_OBFUSCATION_TOOLKIT:

```
UTL_RAW.CAST_TO_RAW(DBMS_OBFUSCATION_TOOLKIT.MD5())
```

An MD5 checksum provides data integrity through hashing and sequencing to ensure that data is not altered or stolen as it is transmitted over a network.

Syntax

```
APEX_ITEM.MD5_CHECKSUM(
    p_value01   IN    VARCHAR2 DEFAULT NULL,
    p_value02   IN    VARCHAR2 DEFAULT NULL,
    p_value03   IN    VARCHAR2 DEFAULT NULL,
    ...
    p_value50   IN    VARCHAR2 DEFAULT NULL,
    p_col_sep   IN    VARCHAR2 DEFAULT '|',
    p_item_id   IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 3–5 describes the parameters available in the `MD5_CHECKSUM` function.

Table 3–5 MD5_CHECKSUM Parameters

Parameter	Description
`p_value01` ... `p_value50`	Fifty available inputs. If no parameters are supplied, the default to NULL
`p_col_sep`	String used to separate `p_value` inputs. Defaults to the pipe symbol (`|`)
`p_item_id`	ID of the HTML form item

Example

This function generates hidden form elements with the name '`fcs`'. The values can subsequently be accessed via the `APEX_APPLICATION.G_FCS array`.

```
SELECT APEX_ITEM.MD5_CHECKSUM(ename,job,sal) md5_cks,
       ename, job, sal
FROM emp
```

MD5_HIDDEN Function

This function is used for lost update detection. Lost update detection ensures data integrity in applications where data can be accessed concurrently.

This function produces a hidden form field and includes 50 inputs. APEX_ITEM.MD5_HIDDEN also produces an MD5 checksum using the Oracle database DBMS_OBFUSCATION_TOOLKIT:

```
UTL_RAW.CAST_TO_RAW(DBMS_OBFUSCATION_TOOLKIT.MD5())
```

An MD5 checksum provides data integrity through hashing and sequencing to ensure that data is not altered or stolen as it is transmitted over a network

Syntax

```
APEX_ITEM.MD5_HIDDEN(
    p_idx       IN    NUMBER,
    p_value01   IN    VARCHAR2 DEFAULT NULL,
    p_value02   IN    VARCHAR2 DEFAULT NULL,
    p_value03   IN    VARCHAR2 DEFAULT NULL,
    ...
    p_value50   IN    VARCHAR2 DEFAULT NULL,
    p_col_sep   IN    VARCHAR2 DEFAULT '|',
    p_item_id   IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 3–6 describes the parameters available in the MD5_HIDDEN function.

Table 3–6 MD5_HIDDEN Parameters

Parameter	Description
p_idx	Indicates the form element to be generated. For example, 1 equals F01 and 2 equals F02. Typically the p_idx parameter is constant for a given column
p_value01	Fifty available inputs. Parameters not supplied default to NULL
...	
p_value50	
p_col_sep	String used to separate p_value inputs. Defaults to the pipe symbol (|)
p_item_id	ID of the HTML form item

Example

The p_idx parameter specifies the FXX form element to be generated. In the following example, 7 generates F07. Also note that an HTML hidden form element will be generated.

```
SELECT APEX_ITEM.MD5_HIDDEN(7,ename,job,sal)md5_h, ename, job, sal
FROM emp
```

POPUP_FROM_LOV Function

This function generates an HTML popup select list from an application shared list of values (LOV). Like other available functions in the APEX_ITEM package, POPUP_FROM_LOV function is designed to generate forms with F01 to F50 form array elements.

Syntax

```
APEX_ITEM.POPUP_FROM_LOV(
    p_idx              IN    NUMBER,
    p_value            IN    VARCHAR2 DEFAULT NULL,
    p_lov_name         IN    VARCHAR2,
    p_width            IN    VARCHAR2 DEFAULT NULL,
    p_max_length       IN    VARCHAR2 DEFAULT NULL,
    p_form_index       IN    VARCHAR2 DEFAULT '0',
    p_escape_html      IN    VARCHAR2 DEFAULT NULL,
    p_max_elements     IN    VARCHAR2 DEFAULT NULL,
    p_attributes       IN    VARCHAR2 DEFAULT NULL,
    p_ok_to_query      IN    VARCHAR2 DEFAULT 'YES',
    p_item_id          IN    VARCHAR2 DEFAULT NULL,
    p_item_label       IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 3–7 describes the some parameters in the POPUP_FROM_LOV function.

Table 3–7 POPUP_FROM_LOV Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, p_idx is a constant for a given column
p_value	Form element current value. This value should be one of the values in the p_lov_name parameter
p_lov_name	Named LOV used for this popup
p_width	Width of the text box
p_max_length	Maximum number of characters that can be entered in the text box
p_form_index	HTML form on the page in which an item is contained. Defaults to 0 and rarely used. Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field that posts to a different Web site). If this form comes before the #FORM_OPEN# substitution string, then its index is zero and the form opened automatically by Oracle Application Express must be referenced as form 1. This functionality supports the JavaScript used in the popup LOV that passes a value back to a form element.

Table 3–7 (Cont.) POPUP_FROM_LOV Parameters

Parameter	Description
p_escape_html	Replacements for special characters that require an escaped equivalent: ■ <lt; for < ■ <gt; for > ■ <amp; for & Range of values is YES and NO. If YES, special characters will be escaped. This parameter is useful if you know your query will return illegal HTML.
p_max_elements	Limit on the number of rows that can be returned by your query. Limits the performance impact of user searches. By entering a value in this parameter, you force the user to search for a narrower set of results.
p_attributes	Additional HTML attributes to use for the form item.
p_ok_to_query	Range of values is YES and NO. If YES, a popup returns first set of rows for the LOV. If NO, a search is initiated to return rows.
p_item_id	ID attribute of the form element.
p_item_label	Invisible label created for the item.

Example

The following example demonstrates a sample query the generates a popup from an LOV named DEPT_LOV.

```
SELECT APEX_ITEM.POPUP_FROM_LOV (1,deptno,'DEPT_LOV') dt
FROM emp
```

POPUP_FROM_QUERY Function

This function generates an HTML popup select list from a query. Like other available functions in the APEX_ITEM package, the POPUP_FROM_QUERY function is designed to generate forms with F01 to F50 form array elements.

Syntax

```
APEX_ITEM.POPUP_FROM_QUERY(

    p_idx              IN    NUMBER,
    p_value            IN    VARCHAR2 DEFAULT NULL,
    p_lov_query        IN    VARCHAR2,
    p_width            IN    VARCHAR2 DEFAULT NULL,
    p_max_length       IN    VARCHAR2 DEFAULT NULL,
    p_form_index       IN    VARCHAR2 DEFAULT '0',
    p_escape_html      IN    VARCHAR2 DEFAULT NULL,
    p_max_elements     IN    VARCHAR2 DEFAULT NULL,
    p_attributes       IN    VARCHAR2 DEFAULT NULL,
    p_ok_to_query      IN    VARCHAR2 DEFAULT 'YES',
    p_item_id          IN    VARCHAR2 DEFAULT NULL,
    p_item_label       IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 3–8 describes the parameters in the POPUP_FROM_QUERY function.

Table 3–8 POPUP_FROM_QUERY Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, p_idx is a constant for a given column.
p_value	Form element current value. This value should be one of the values in the p_lov_query parameter.
p_lov_query	SQL query that is expected to select two columns (a display column and a return column). For example: `SELECT dname, deptno FROM dept`
p_width	Width of the text box.
p_max_length	Maximum number of characters that can be entered in the text box.
p_form_index	HTML form on the page in which an item is contained. Defaults to 0 and rarely used. Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field that posts to a different Web site). If this form comes before the #FORM_OPEN# substitution string, then its index is zero and the form opened automatically by Oracle Application Express must be referenced as form 1. This functionality supports the JavaScript used in the popup LOV that passes a value back to a form element.

Table 3–8 (Cont.) POPUP_FROM_QUERY Parameters

Parameter	Description
p_escape_html	Replacements for special characters that require an escaped equivalent. ■ <lt; for < ■ <gt; for > ■ <amp; for & Range of values is YES and NO. If YES, special characters will be escaped. This parameter is useful if you know your query will return illegal HTML.
p_max_elements	Limit on the number of rows that can be returned by your query. Limits the performance impact of user searches. By entering a value in this parameter, you force the user to search for a narrower set of results.
p_attributes	Additional HTML attributes to use for the form item.
p_ok_to_query	Range of values is YES and NO. If YES, a popup returns the first set of rows for the LOV. If NO, a search is initiated to return rows.
p_item_id	ID attribute of the form element.
p_item_label	Invisible label created for the item.

Example

The following example demonstrates a sample query the generates a popup select list from the emp table.

```
SELECT APEX_ITEM.POPUP_FROM_QUERY (1,deptno,'SELECT dname, deptno FROM dept') dt
FROM emp
```

POPUPKEY_FROM_LOV Function

This function generates a popup key select list from a shared list of values (LOV). Similar to other available functions in the APEX_ITEM package, the POPUPKEY_FROM_LOV function is designed to generate forms with F01 to F50 form array elements.

Syntax

```
APEX_ITEM.POPUPKEY_FROM_LOV(
    p_idx              IN    NUMBER,
    p_value            IN    VARCHAR2 DEFAULT NULL,
    p_lov_name         IN    VARCHAR2,
    p_width            IN    VARCHAR2 DEFAULT NULL,
    p_max_length       IN    VARCHAR2 DEFAULT NULL,
    p_form_index       IN    VARCHAR2 DEFAULT '0',
    p_escape_html      IN    VARCHAR2 DEFAULT NULL,
    p_max_elements     IN    VARCHAR2 DEFAULT NULL,
    p_attributes       IN    VARCHAR2 DEFAULT NULL,
    p_ok_to_query      IN    VARCHAR2 DEFAULT 'YES',
    p_item_id          IN    VARCHAR2 DEFAULT NULL,
    p_item_label       IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Although the text field associated with the popup displays in the first column in the LOV query, the actual value is specified in the second column in the query.

Parameters

Table 3–9 describes the some parameters in the POPUPKEY_FROM_LOV function.

Table 3–9 POPUPKEY_FROM_LOV Parameters

Parameter	Description
p_idx	Identifies a form element name. For example, 1 equals F01 and 2 equals F02. Typically, p_idx is a constant for a given column Because of the behavior of POPUPKEY_FROM_QUERY, the next index value should be p_idx + 1. For example: SELECT APEX_ITEM.POPUPKEY_FROM_LOV (1,deptno,'DEPT') dt, APEX_ITEM.HIDDEN(3,empno) eno
p_value	Indicates the current value. This value should be one of the values in the P_LOV_NAME parameter.
p_lov_name	Identifies a named LOV used for this popup.
p_width	Width of the text box.
p_max_length	Maximum number of characters that can be entered in the text box.
p_form_index	HTML form on the page in which an item is contained. Defaults to 0 and rarely used. Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field that posts to a different Web site). If this form comes before the #FORM_OPEN# substitution string, then its index is zero and the form opened automatically by Oracle Application Express must be referenced as form 1. This functionality supports the JavaScript used in the popup LOV that passes a value back to a form element.

Table 3–9 (Cont.) POPUPKEY_FROM_LOV Parameters

Parameter	Description
p_escape_html	Replacements for special characters that require an escaped equivalent. ■ < for < ■ > for > ■ & for & This parameter is useful if you know your query will return illegal HTML.
p_max_elements	Limit on the number of rows that can be returned by your query. Limits the performance impact of user searches. By entering a value in this parameter, you force the user to search for a narrower set of results.
p_attributes	Additional HTML attributes to use for the form item.
p_ok_to_query	Range of values is YES and NO. If YES, a popup returns the first set of rows for the LOV. If NO, a search is initiated to return rows.
p_item_id	HTML attribute ID for the <input> tag
p_item_label	Invisible label created for the item

Example

The following example demonstrates how to generate a popup key select list from a shared list of values (LOV).

```
SELECT APEX_ITEM.POPUPKEY_FROM_LOV (1,deptno,'DEPT') dt
FROM emp
```

POPUPKEY_FROM_QUERY Function

This function generates a popup key select list from a SQL query. Similar to other available functions in the APEX_ITEM package, the POPUPKEY_FROM_QUERY function is designed to generate forms with F01 to F50 form array elements.

Syntax

```
APEX_ITEM.POPUPKEY_FROM_QUERY(
    p_idx              IN    NUMBER,
    p_value            IN    VARCHAR2 DEFAULT NULL,
    p_lov_query        IN    VARCHAR2,
    p_width            IN    VARCHAR2 DEFAULT NULL,
    p_max_length       IN    VARCHAR2 DEFAULT NULL,
    p_form_index       IN    VARCHAR2 DEFAULT '0',
    p_escape_html      IN    VARCHAR2 DEFAULT NULL,
    p_max_elements     IN    VARCHAR2 DEFAULT NULL,
    p_attributes       IN    VARCHAR2 DEFAULT NULL,
    p_ok_to_query      IN    VARCHAR2 DEFAULT 'YES',
    p_item_id          IN    VARCHAR2 DEFAULT NULL,
    p_item_label       IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 3–10 describes the some parameters in the POPUPKEY_FROM_QUERY function.

Table 3–10 POPUPKEY_FROM_QUERY Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, p_idx is a constant for a given column. Because of the behavior of POPUPKEY_FROM_QUERY, the next index value should be p_idx + 1. For example: `SELECT APEX_ITEM.POPUPKEY_FROM_QUERY (1,deptno,'SELECT dname, deptno FROM dept') dt, APEX_ITEM.HIDDEN(3,empno) eno`
p_value	Form element current value. This value should be one of the values in the P_LOV_QUERY parameter.
p_lov_query	LOV query used for this popup.
p_width	Width of the text box.
p_max_length	Maximum number of characters that can be entered in the text box.
p_form_index	HTML form on the page in which an item is contained. Defaults to 0 and rarely used. Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field that posts to a different Web site). If this form comes before the #FORM_OPEN# substitution string, then its index is zero and the form opened automatically by Oracle Application Express must be referenced as form 1. This functionality supports the JavaScript used in the popup LOV that passes a value back to a form element.

Table 3–10 (Cont.) POPUPKEY_FROM_QUERY Parameters

Parameter	Description
p_escape_html	Replacements for special characters that require an escaped equivalent. ■ < for < ■ > for > ■ & for & This parameter is useful if you know your query will return illegal HTML.
p_max_elements	Limit on the number of rows that can be returned by your query. Limits the performance impact of user searches. By entering a value in this parameter, you force the user to search for a narrower set of results.
p_attributes	Additional HTML attributes to use for the form item.
p_ok_to_query	Range of values is YES and NO. If YES, a popup returns first set of rows for the LOV. If NO, a search is initiated to return rows.
p_item_id	ID attribute of the form element.
p_item_label	Invisible label created for the item.

Example

The following example demonstrates how to generate a popup select list from a SQL query.

```
SELECT APEX_ITEM.POPUPKEY_FROM_QUERY (1,deptno,'SELECT dname, deptno FROM dept')
dt
FROM emp
```

RADIOGROUP Function

This function generates a radio group from a SQL query.

Syntax

```
APEX_ITEM.RADIOGROUP(
    p_idx              IN    NUMBER,
    p_value            IN    VARCHAR2 DEFAULT NULL,
    p_selected_value   IN    VARCHAR2 DEFAULT NULL,
    p_display          IN    VARCHAR2 DEFAULT NULL,
    p_attributes       IN    VARCHAR2 DEFAULT NULL,
    p_onblur           IN    VARCHAR2 DEFAULT NULL,
    p_onchange         IN    VARCHAR2 DEFAULT NULL,
    p_onfocus          IN    VARCHAR2 DEFAULT NULL,
    p_item_id          IN    VARCHAR2 DEFAULT NULL,
    p_item_label       IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 3–11 describes the parameters available in the `RADIOGROUP` function.

Table 3–11 RADIOGROUP Parameters

Parameter	Description
`p_idx`	Number that determines which `APEX_APPLICATION` global variable will be used. Valid range of values is `1` to `50`.For example `1` creates `F01` and `2` creates `F02`.
`p_value`	Value of the radio group.
`p_selected_value`	Value that should be selected.
`p_display`	Text to display next to the radio option.
`p_attributes`	Extra HTML parameters you want to add.
`p_onblur`	JavaScript to execute in the `onBlur` event.
`p_onchange`	JavaScript to execute in the `onChange` event.
`p_onfocus`	JavaScript to execute in the `onFocus` event.
`p_item_id`	HTML attribute ID for the <input> tag
`p_item_label`	Invisible label created for the item

Example

The following example demonstrates how to select department 20 from the `emp` table as a default in a radio group.

```
SELECT APEX_ITEM.RADIOGROUP (1,deptno,'20',dname) dt
FROM   dept
ORDER  BY 1
```

SELECT_LIST Function

This function dynamically generates a static select list. Similar to other functions available in the APEX_ITEM package, these select list functions are designed to generate forms with F01 to F50 form array elements.

Syntax

```
APEX_ITEM.SELECT_LIST(
    p_idx           IN   NUMBER,
    p_value         IN   VARCHAR2 DEFAULT NULL,
    p_list_values   IN   VARCHAR2 DEFAULT NULL,
    p_attributes    IN   VARCHAR2 DEFAULT NULL,
    p_show_null     IN   VARCHAR2 DEFAULT 'NO',
    p_null_value    IN   VARCHAR2 DEFAULT '%NULL%',
    p_null_text     IN   VARCHAR2 DEFAULT '%',
    p_item_id       IN   VARCHAR2 DEFAULT NULL,
    p_item_label    IN   VARCHAR2 DEFAULT NULL,
    p_show_extra    IN   VARCHAR2 DEFAULT 'YES')
    RETURN VARCHAR2;
```

Parameters

Table 3–12 describes the parameters available in the SELECT_LIST function.

Table 3–12 SELECT_LIST Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically the P_IDX parameter is constant for a given column.
p_value	Current value. This value should be a value in the P_LIST_VALUES parameter.
p_list_values	List of static values separated by commas. Displays values and returns values that are separated by semicolons. Note that this is only available in the SELECT_LIST function.
p_attributes	Extra HTML parameters you want to add.
p_show_null	Extra select option to enable the NULL selection. Range of values is YES and NO.
p_null_value	Value to be returned when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_null_text	Value to be displayed when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_item_id	HTML attribute ID for the <input> tag.
p_item_label	Invisible lable created for the item.
p_show_extra	Shows the current value even if the value of p_value is not located in the select list.

Example

The following example demonstrates a static select list that displays Yes, returns Y, defaults to Y, and generates a F01 form item.

```
SELECT APEX_ITEM.SELECT_LIST(1,'Y','Yes;Y,No;N')yn
```

```
FROM emp
```

The following example demonstrates the use of `APEX_ITEM.SELECT_LIST` to generate a static select list where:

- A form array element `F03` will be generated (`p_idx` parameter).
- The initial value for each element will be equal to the value for `deptno` for the row from `emp` (`p_value` parameter).
- The select list will contain 4 options (`p_list_values` parameter).
- The text within the select list will display in red (`p_attributes` parameter).
- A null option will be displayed (`p_show_null`) and this option will display `-Select-` as the text (`p_null_text` parameter).
- An HTML ID attribute will be generated for each row, where `#ROWNUM#` will be substituted for the current row `rownum` (`p_item_id` parameter). (So an ID of '`f03_4`' will be generated for row 4.)
- A HTML label element will be generated for each row (`p_item_label` parameter).
- The current value for `deptno` will be displayed, even if it is not contained with the list of values passed in the `p_list_values` parameter (`p_show_extra` parameter).

```
SELECT  empno "Employee #",
    ename "Name",
    APEX_ITEM.SELECT_LIST(
        p_idx           =>  3,
        p_value         =>  deptno,
        p_list_values   =>  'ACCOUNTING;10,RESEARCH;20,SALES;30,OPERATIONS;40',
        p_attributes    =>  'style="color:red;"',
        p_show_null     =>  'YES',
        p_null_value    =>  NULL,
        p_null_text     =>  '-Select-',
        p_item_id       =>  'f03_#ROWNUM#',
        p_item_label    =>  'Label for f03_#ROWNUM#',
        p_show_extra    =>  'YES') "Department"
  FROM  emp;
```

SELECT_LIST_FROM_LOV Function

This function dynamically generates select lists from a shared list of values (LOV). Similar to other functions available in the APEX_ITEM package, these select list functions are designed to generate forms with F01 to F50 form array elements.

Syntax

```
APEX_ITEM.SELECT_LIST_FROM_LOV(
    p_idx           IN   NUMBER,
    p_value         IN   VARCHAR2 DEFAULT NULL,
    p_lov           IN   VARCHAR2,
    p_attributes    IN   VARCHAR2 DEFAULT NULL,
    p_show_null     IN   VARCHAR2 DEFAULT 'YES',
    p_null_value    IN   VARCHAR2 DEFAULT '%NULL%',
    p_null_text     IN   VARCHAR2 DEFAULT '%',
    p_item_id       IN   VARCHAR2 DEFAULT NULL,
    p_item_label    IN   VARCHAR2 DEFAULT NULL,
    p_show_extra    IN   VARCHAR2 DEFAULT 'YES')
    RETURN VARCHAR2;
```

Parameters

Table 3–13 describes the parameters available in the SELECT_LIST_FROM_LOV function.

Table 3–13 SELECT_LIST_FROM_LOV Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, the p_idx parameter is constant for a given column.
p_value	Current value. This value should be a value in the p_lov parameter.
p_lov	Text name of an application list of values. This list of values must be defined in your application. This parameter is used only by the select_list_from_lov function.
p_attributes	Extra HTML parameters you want to add.
p_show_null	Extra select option to enable the NULL selection. Range of values is YES and NO.
p_null_value	Value to be returned when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_null_text	Value to be displayed when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_item_id	HTML attribute ID for the <select> tag.
p_item_label	Invisible label created for the item.
p_show_extra	Shows the current value even if the value of p_value is not located in the select list.

Example

The following example demonstrates a select list based on an LOV defined in the application.

```
SELECT APEX_ITEM.SELECT_LIST_FROM_LOV(2,job,'JOB_FLOW_LOV')job
FROM emp
```

SELECT_LIST_FROM_LOV_XL Function

This function dynamically generates very large select lists (greater than 32K) from a shared list of values (LOV). Similar to other functions available in the APEX_ITEM package, these select list functions are designed to generate forms with F01 to F50 form array elements. This function is the same as SELECT_LIST_FROM_LOV, but its return value is CLOB. This enables you to use it in SQL queries where you need to handle a column value longer than 4000 characters.

Syntax

```
APEX_ITEM.SELECT_LIST_FROM_LOV_XL(
    p_idx           IN   NUMBER,
    p_value         IN   VARCHAR2 DEFAULT NULL,
    p_lov           IN   VARCHAR2,
    p_attributes    IN   VARCHAR2 DEFAULT NULL,
    p_show_null     IN   VARCHAR2 DEFAULT 'YES',
    p_null_value    IN   VARCHAR2 DEFAULT '%NULL%',
    p_null_text     IN   VARCHAR2 DEFAULT '%',
    p_item_id       IN   VARCHAR2 DEFAULT NULL,
    p_item_label    IN   VARCHAR2 DEFAULT NULL,
    p_show_extra    IN   VARCHAR2 DEFAULT 'YES')
    RETURN CLOB;
```

Parameters

Table 3–14 describes the parameters available in the SELECT_LIST_FROM_LOV_XL function.

Table 3–14 SELECT_LIST_FROM_LOV_XL Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, the p_idx parameter is constant for a given column.
p_value	Current value. This value should be a value in the p_lov parameter.
p_lov	Text name of a list of values. This list of values must be defined in your application. This parameter is used only by the select_list_from_lov function.
p_attributes	Extra HTML parameters you want to add.
p_show_null	Extra select option to enable the NULL selection. Range of values is YES and NO.
p_null_value	Value to be returned when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_null_text	Value to be displayed when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_item_id	HTML attribute ID for the <select> tag.
p_item_label	Invisible label created for the item.
p_show_extra	Shows the current value even if the value of p_value is not located in the select list.

Example

The following example demonstrates how to create a select list based on an LOV defined in the application.

```
SELECT APEX_ITEM.SELECT_LIST_FROM_LOV_XL(2,job,'JOB_FLOW_LOV')job
FROM emp
```

SELECT_LIST_FROM_QUERY Function

This function dynamically generates a select list from a query. Similar to other functions available in the APEX_ITEM package, these select list functions are designed to generate forms with F01 to F50 form array elements.

Syntax

```
APEX_ITEM.SELECT_LIST_FROM_QUERY(
    p_idx           IN    NUMBER,
    p_value         IN    VARCHAR2 DEFAULT NULL,
    p_query         IN    VARCHAR2,
    p_attributes    IN    VARCHAR2 DEFAULT NULL,
    p_show_null     IN    VARCHAR2 DEFAULT 'YES',
    p_null_value    IN    VARCHAR2 DEFAULT '%NULL%',
    p_null_text     IN    VARCHAR2 DEFAULT '%',
    p_item_id       IN    VARCHAR2 DEFAULT NULL,
    p_item_label    IN    VARCHAR2 DEFAULT NULL,
    p_show_extra    IN    VARCHAR2 DEFAULT 'YES')
    RETURN VARCHAR2;
```

Parameters

Table 3–15 describes the parameters available in the SELECT_LIST_FROM_QUERY function.

Table 3–15 SELECT_LIST_FROM_QUERY Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, the p_idx parameter is constant for a given column.
p_value	Current value. This value should be a value in the p_query parameter.
p_query	SQL query that is expected to select two columns, a display column, and a return column. For example: `SELECT dname, deptno FROM dept` Note that this is used only by the SELECT_LIST_FROM_QUERY function. Also note, if only one column is specified in the select clause of this query, the value for this column will be used for both display and return purposes.
p_attributes	Extra HTML parameters you want to add.
p_show_null	Extra select option to enable the NULL selection. Range of values is YES and NO.
p_null_value	Value to be returned when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_null_text	Value to be displayed when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_item_id	HTML attribute ID for the <select> tag.
p_item_label	Invisible label created for the item.
p_show_extra	Show the current value even if the value of p_value is not located in the select list.

Example

The following example demonstrates a select list based on a SQL query.

```
SELECT APEX_ITEM.SELECT_LIST_FROM_QUERY(3,job,'SELECT DISTINCT job FROM emp')job
FROM emp
```

SELECT_LIST_FROM_QUERY_XL Function

This function is the same as SELECT_LIST_FROM_QUERY, but its return value is a CLOB. This allows its use in SQL queries where you need to handle a column value longer than 4000 characters. Similar to other functions available in the APEX_ITEM package, these select list functions are designed to generate forms with F01 to F50 form array elements.

Syntax

```
APEX_ITEM.SELECT_LIST_FROM_QUERY_XL(
    p_idx           IN    NUMBER,
    p_value         IN    VARCHAR2 DEFAULT NULL,
    p_query         IN    VARCHAR2,
    p_attributes    IN    VARCHAR2 DEFAULT NULL,
    p_show_null     IN    VARCHAR2 DEFAULT 'YES',
    p_null_value    IN    VARCHAR2 DEFAULT '%NULL%',
    p_null_text     IN    VARCHAR2 DEFAULT '%',
    p_item_id       IN    VARCHAR2 DEFAULT NULL,
    p_item_label    IN    VARCHAR2 DEFAULT NULL,
    p_show_extra    IN    VARCHAR2 DEFAULT 'YES')
    RETURN CLOB;
```

Parameters

Table 3–16 describes the parameters available in the SELECT_LIST_FROM_QUERY_XL function.

Table 3–16 SELECT_LIST_FROM_QUERY_XL Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically the p_idx parameter is constant for a given column.
p_value	Current value. This value should be a value in the p_query parameter.
p_query	SQL query that is expected to select two columns, a display column, and a return column. For example: `SELECT dname, deptno FROM dept` Note that this is used only by the SELECT_LIST_FROM_QUERY_XL function. Also note, if only one column is specified in the select clause of this query, the value for this column will be used for both display and return purposes.
p_attributes	Extra HTML parameters you want to add.
p_show_null	Extra select option to enable the NULL selection. Range of values is YES and NO.
p_null_value	Value to be returned when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_null_text	Value to be displayed when a user selects the NULL option. Only relevant when p_show_null equals YES.
p_item_id	HTML attribute ID for the <select> tag.
p_item_label	Invisible label created for the item.

Table 3–16 (Cont.) SELECT_LIST_FROM_QUERY_XL Parameters

Parameter	Description
p_show_extra	Show the current value even if the value of p_value is not located in the select list.

Example

The following example demonstrates a select list based on a SQL query.

```
SELECT APEX_ITEM.SELECT_LIST_FROM_QUERY_XL(3,job,'SELECT DISTINCT job FROM
emp')job
FROM emp
```

TEXT Function

This function generates text fields (or text input form items) from a SQL query.

Syntax

```
APEX_ITEM.TEXT(
    p_idx         IN    NUMBER,
    p_value       IN    VARCHAR2 DEFAULT NULL,
    p_size        IN    NUMBER DEFAULT NULL,
    p_maxlength   IN    NUMBER DEFAULT NULL,
    p_attributes  IN    VARCHAR2 DEFAULT NULL,
    p_item_id     IN    VARCHAR2 DEFAULT NULL,
    p_item_label  IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 3–17 describes the parameters available in the TEXT function.

Table 3–17 TEXT Parameters

Parameter	Description
p_idx	Number to identify the item you want to generate. The number will determine which G_FXX global is populated. **See Also:** "APEX_APPLICATION" on page 4-1
p_value	Value of a text field item.
p_size	Controls HTML tag attributes (such as disabled).
p_maxlength	Maximum number of characters that can be entered in the text box.
p_attributes	Extra HTML parameters you want to add.
p_item_id	HTML attribute ID for the <input> tag.
p_item_label	Invisible label created for the item.

Example

The following sample query demonstrates how to generate one update field for each row. Note that the ename, sal, and comm columns use the APEX_ITEM.TEXT function to generate an HTML text field for each row. Also, notice that each item in the query is passed a unique p_idx parameter to ensure that each column is stored in its own array.

```
SELECT
  empno,
  APEX_ITEM.HIDDEN(1,empno)||
  APEX_ITEM.TEXT(2,ename) ename,
  APEX_ITEM.TEXT(3,job) job,
  mgr,
  APEX_ITEM.DATE_POPUP(4,rownum,hiredate,'dd-mon-yyyy') hiredate,
  APEX_ITEM.TEXT(5,sal) sal,
  APEX_ITEM.TEXT(6,comm) comm,
  deptno
FROM emp
ORDER BY 1
```

TEXTAREA Function

This function creates text areas.

Syntax

```
APEX_ITEM.TEXTAREA(
    p_idx         IN    NUMBER,
    p_value       IN    VARCHAR2 DEFAULT NULL,
    p_rows        IN    NUMBER DEAULT 40,
    p_cols        IN    NUMBER DEFAULT 4,
    p_attributes  IN    VARCHAR2 DEFAULT NULL,
    p_item_id     IN    VARCHAR2 DEFAULT NULL,
    p_item_label  IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 3–18 describes the parameters available in the `TEXTAREA` function.

Table 3–18 TEXTAREA Parameters

Parameter	Description
`p_idx`	Number to identify the item you want to generate. The number will determine which `G_FXX` global is populated. **See Also:** "APEX_APPLICATION" on page 4-1
`p_value`	Value of the text area item.
`p_rows`	Height of the text area (HTML rows attribute)
`p_cols`	Width of the text area (HTML column attribute).
`p_attributes`	Extra HTML parameters you want to add.
`p_item_id`	HTML attribute ID for the `<textarea>` tag.
`p_item_label`	Invisible label created for the item.

Example

The following example demonstrates how to create a text area based on a SQL query.

```
SELECT APEX_ITEM.TEXTAREA(3,ename,5,80) a
FROM emp
```

TEXT_FROM_LOV Function

Use this function to display an item as text, deriving the display value of the named LOV.

Syntax

```
APEX_ITEM.TEXT_FROM_LOV (
    p_value       IN    VARCHAR2 DEFAULT NULL,
    p_lov         IN    VARCHAR2,
    p_null_text   IN    VARCHAR2 DEFAULT '%')
    RETURN VARCHAR2;
```

Parameters

Table 3–19 describes the parameters available in the TEXT_FROM_LOV function.

Table 3–19 TEXT_FROM_LOV Parameters

Parameter	Description
p_value	Value of a field item. Note that if p_value is not located in the list of values, p_null_text is value displayed.
p_lov	Text name of a shared list of values. This list of values must be defined in your application.
p_null_text	Value displayed when the value of the field item is NULL.

Example

The following example demonstrates how to derive the display value from a named LOV (EMPNO_ENAME_LOV).

```
SELECT APEX_ITEM.TEXT_FROM_LOV(empno,'EMPNO_ENAME_LOV') c FROM emp
```

TEXT_FROM_LOV_QUERY Function

Use this function to display an item as text, deriving the display value from a list of values query.

Syntax

```
APEX_ITEM.TEXT_FROM_LOV_QUERY (
    p_value       IN    VARCHAR2 DEFAULT NULL,
    p_query       IN    VARCHAR2,
    p_null_text   IN    VARCHAR2 DEFAULT '%')
    RETURN VARCHAR2;
```

Parameters

Table 3–20 describes the parameters available in the `TEXT_FROM_LOV_QUERY` function.

Table 3–20 TEXT_FROM_LOV_QUERY Parameters

Parameter	Description
`p_value`	Value of a field item.
`p_query`	SQL query that is expected to select two columns, a display column and a return column. For example: `SELECT dname, deptno FROM dept` Note if only one column is specified in the select clause of this query, the value for this column will be used for both display and return purposes.
`p_null_text`	Value to be displayed when the value of the field item is NULL or a corresponding entry is not located for the value `p_value` in the list of values query.

Example

The following example demonstrates how to derive the display value from a query.

```
SELECT APEX_ITEM.TEXT_FROM_LOV_QUERY(empno,'SELECT ename, empno FROM emp') c from
emp
```

4

APEX_APPLICATION

The APEX_APPLICATION package is a PL/SQL package that implements the Oracle Application Express rendering engine. You can use this package to take advantage of a number of global variables. Table 4–1 describes the global variables available in the APEX_APPLICATION package.

Table 4–1 Global Variables Available in APEX_APPLICATION

Global Variable	Description
G_USER	Specifies the currently logged in user.
G_FLOW_ID	Specifies the ID of the currently running application.
G_FLOW_STEP_ID	Specifies the ID of the currently running page.
G_FLOW_OWNER	Specifies the schema to parse for the currently running application.
G_REQUEST	Specifies the value of the request variable most recently passed to or set within the show or accept modules.
G_BROWSER_LANGUAGE	Refers to the Web browser's current language preference.
G_DEBUG	Refers to whether debugging is currently switched on or off. Valid values for the DEBUG flag are 'Yes' or 'No'. Turning debug on shows details about application processing.
G_HOME_LINK	Refers to the home page of an application. The Application Express engine will redirect to this location if no page is given and if no alternative page is dictated by the authentication scheme's logic.
G_LOGIN_URL	Can be used to display a link to a login page for users that are not currently logged in.
G_IMAGE_PREFIX	Refers to the virtual path the web server uses to point to the images directory distributed with Oracle Application Express.
G_FLOW_SCHEMA_OWNER	Refers to the owner of the Application Express schema.
G_PRINTER_FRIENDLY	Refers to whether or not the Application Express engine is running in print view mode. This setting can be referenced in conditions to eliminate elements not desired in a printed document from a page.
G_PROXY_SERVER	Refers to the application attribute 'Proxy Server'.
G_SYSDATE	Refers to the current date on the database server. this uses the DATE DATATYPE.
G_PUBLIC_USER	Refers to the Oracle schema used to connect to the database through the database access descriptor (DAD).
G_GLOBAL_NOTIFICATION	Specifies the application's global notification attribute.

Topics:

- Referencing Arrays
- Referencing Values Within an On Submit Process
- Converting an Array to a Single Value
- HELP Procedure

Referencing Arrays

Items are typically HTML form elements such as text fields, select lists, and check boxes. When you create a new form item using a wizard, the wizard uses a standard naming format. The naming format provides a handle so you can retrieve the value of the item later on.

If you need to create your own items, you can access them after a page is submitted by referencing `APEX_APPLICATION.G_F01` to `APEX_APPLICATION.G_F50` arrays. You can create your own HTML form fields by providing the input parameters using the format `F01`, `F02`, `F03` and so on. You can create up to 50 input parameters ranging from `F01` to `F50`, for example:

```
<INPUT TYPE="text" NAME="F01" SIZE="32" MAXLENGTH="32" VALUE="some value">

<TEXTAREA NAME="F02" ROWS=4 COLS=90 WRAP="VIRTUAL">this is the example of a text
area.</TEXTAREA>

<SELECT NAME="F03" SIZE="1">
<OPTION VALUE="abc">abc
<OPTION VALUE="123">123
</SELECT>
```

Because the `F01` to `F50` input items are declared as PL/SQL arrays, you can have multiple items named the same value. For example:

```
<INPUT TYPE="text" NAME="F01" SIZE="32" MAXLENGTH="32" VALUE="array element 1">
<INPUT TYPE="text" NAME="F01" SIZE="32" MAXLENGTH="32" VALUE="array element 2">
<INPUT TYPE="text" NAME="F01" SIZE="32" MAXLENGTH="32" VALUE="array element 3">
```

Note that following PL/SQL code produces the same HTML as show in the previous example.

```
FOR i IN 1..3 LOOP
APEX_ITEM.TEXT(P_IDX        => 1,
   p_value      =>'array element '||i ,
   p_size       =>32,
   p_maxlength  =>32);
END LOOP;
```

Referencing Values Within an On Submit Process

You can reference the values posted by an HTML form using the PL/SQL variable APEX_APPLICATION.G_F01 to APEX_APPLICATION.G_F50. Because this element is an array, you can reference values directly, for example:

```
FOR i IN 1..APEX_APPLICATION.G_F01.COUNT LOOP
    htp.p('element '||I||' has a value of '||APEX_APPLICATION.G_F01(i));
END LOOP;
```

Note that check boxes displayed using APEX_ITEM.CHECKBOX will only contain values in the APEX_APPLICATION arrays for those rows which are checked. Unlike other items (TEXT, TEXTAREA, and DATE_POPUP) which can contain an entry in the corresponding APEX_APPLICATION array for every row submitted, a check box will only have an entry in the APEX_APPLICATION array if it is selected.

Converting an Array to a Single Value

You can also use Oracle Application Express public utility functions to convert an array into a single value. The resulting string value is a colon-separated list of the array element values. For example:

```
htp.p(APEX_UTIL.TABLE_TO_STRING(APEX_APPLICATION.G_F01));
```

This function enables you to reference `G_F01` to `G_F50` values in an application process that performs actions on data. The following sample process demonstrates how values are inserted into a table:

```
INSERT INTO my_table (my_column) VALUES
APEX_UTIL.TABLE_TO_STRING(APEX_APPLICATION.G_F01)
```

HELP Procedure

This function outputs page and item level help text as formatted HTML and can be used to customize how help information is displayed in your application.

Syntax

```
APEX_APPLICATION.HELP (
    p_request        IN VARCHAR2 DEFAULT NULL,
    p_flow_id        IN VARCHAR2 DEFAULT NULL,
    p_flow_step_id   IN VARCHAR2 DEFAULT NULL,
    p_show_item_help IN VARCHAR2 DEFAULT 'YES',
    p_show_regions   IN VARCHAR2 DEFAULT 'YES',
    p_before_page_html     IN VARCHAR2 DEFAULT '<p>',
    p_after_page_html      IN VARCHAR2 DEFAULT NULL,
    p_before_region_html   IN VARCHAR2 DEFAULT NULL,
    p_after_region_html    IN VARCHAR2 DEFAULT '</td></tr></table></p>',
    p_before_prompt_html   IN VARCHAR2 DEFAULT '<p><b>',
    p_after_prompt_html    IN VARCHAR2 DEFAULT '</b></p>: ',
    p_before_item_html     IN VARCHAR2 DEFAULT NULL,
    p_after_item_html      IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 4–2 describes the parameters available in the HELP procedure.

Table 4–2 HELP Parameters

Parameter	Description
p_request	Not used.
p_flow_id	The application ID that contains the page or item level help you want to output.
p_flow_step_id	The page ID that contains the page or item level help you want to display.
p_show_item_help	Flag to determine if item level help is output. If this parameter is supplied, the value must be either 'YES' or 'NO', if not the default value will be 'YES'.
p_show_regions	Flag to determine if region headers are output (for regions containing page items). If this parameter is supplied, the value must be either 'YES' or 'NO', if not the default value will be 'YES'.
p_before_page_html	Use this parameter to include HTML between the page level help text and item level help text.
p_after_page_html	Use this parameter to include HTML at the bottom of the output, after all other help.
p_before_region_html	Use this parameter to include HTML before every region section. Note this parameter is ignored if p_show_regions is set to 'NO'.
p_after_region_html	Use this parameter to include HTML after every region section. Note this parameter is ignored if p_show_regions is set to 'NO'.
p_before_prompt_html	Use this parameter to include HTML before every item label for item level help. Note this parameter is ignored if p_show_item_help is set to 'NO'.

Table 4–2 (Cont.) HELP Parameters

Parameter	Description
p_after_prompt_html	Use this parameter to include HTML after every item label for item level help. Note this parameter is ignored if p_show_item_help is set to 'NO'.
p_before_item_html	Use this parameter to include HTML before every item help text for item level help. Note this parameter is ignored if p_show_item_help is set to 'NO'.
p_after_item_html	Use this parameter to include HTML after every item help text for item level help. Note this parameter is ignored if p_show_item_help is set to 'NO'.

Example

The following example shows how to use the APEX_APPLICATION.HELP procedure to customize how help information is displayed.

In this example, the p_flow_step_id parameter is set to :REQUEST, which means that a page ID specified in the REQUEST section of the URL will be used to control which page's help information to display (see note after example for full details on how this can be achieved).

Also, the help display has been customized so that the region sub-header now has a different color (through the p_before_region_html parameter) and also the ':' has been removed that appeared by default after every item prompt (through the p_after_prompt_html parameter).

```
APEX_APPLICATION.HELP(
    p_flow_id => :APP_ID,
    p_flow_step_id => :REQUEST,
    p_before_region_html => '<p><br/><table bgcolor="#A3BED8"
width="100%"><tr><td><b>',
    p_after_prompt_html  => '</b></p>  ');
```

In order to implement this type of call in your application, you can do the following:

1. Create a page that will be your application help page.
2. Create a region of type 'PL/SQL Dynamic Content' and add the APEX_APPLICATION.HELP call as PL/SQL Source.
3. Then you can add a 'Navigation Bar' link to this page, ensuring that the REQUEST value set in the link is &APP_PAGE_ID.

5

APEX_CUSTOM_AUTH

You can use the `APEX_CUSTOM_AUTH` package to perform various operations related to authentication and session management.

Topics:

- APPLICATION_PAGE_ITEM_EXISTS Function
- CURRENT_PAGE_IS_PUBLIC Function
- DEFINE_USER_SESSION Procedure
- GET_COOKIE_PROPS Procedure
- GET_LDAP_PROPS Procedure
- GET_NEXT_SESSION_ID Function
- GET_SECURITY_GROUP_ID Function
- GET_SESSION_ID Function
- GET_SESSION_ID_FROM_COOKIE Function
- GET_USER Function
- GET_USERNAME Function
- IS_SESSION_VALID Function
- LOGIN Procedure
- LOGOUT Procedure
- POST_LOGIN Procedure
- SESSION_ID_EXISTS Function
- SET_SESSION_ID Procedure
- SET_SESSION_ID_TO_NEXT_VALUE Procedure
- SET_USER Procedure

APPLICATION_PAGE_ITEM_EXISTS Function

This function checks for the existence of page-level item within the current page of an application. This function requires the parameter `p_item_name`. This function returns a Boolean value (true or false).

Syntax

```
APEX_CUSTOM_AUTH.APPLICATION_PAGE_ITEM_EXISTS(
    p_item_name   IN    VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 5–1 describes the parameters available in the APPLICATION_PAGE_ITEM_EXISTS function.

Table 5–1 APPLICATION_PAGE_ITEM_EXISTS Parameters

Parameter	Description
`p_item_name`	The name of the page-level item.

Example

The following example checks for the existance of a page-level item, `ITEM_NAME`, within the current page of the application.

```
DECLARE
    L_VAL BOOLEAN;
BEGIN
    VAL := APEX_CUSTOM_AUTH.APPLICATION_PAGE_ITEM_EXISTS(:ITEM_NAME);
    IF L_VAL THEN
        htp.p('Item Exists');
    ELSE
        htp.p('Does not Exist');
    END IF;
END;
```

CURRENT_PAGE_IS_PUBLIC Function

This function checks whether the current page's authentication attribute is set to **Page Is Public** and returns a Boolean value (true or false)

> **See Also:** "Editing Page Attributes" in *Oracle Application Express Application Builder User's Guide*.

Syntax

```
APEX_CUSTOM_AUTH.CURRENT_PAGE_IS_PUBLIC
RETURN BOOLEAN;
```

Example

The following example checks whether the current page in an application is public.

```
DECLARE
    L_VAL BOOLEAN;
BEGIN
    L_VAL := APEX_CUSTOM_AUTH.CURRENT_PAGE_IS_PUBLIC;
    IF L_VAL THEN
        htp.p('Page is Public');
    ELSE
        htp.p('Page is not Public');
    END IF;
END;
```

DEFINE_USER_SESSION Procedure

This procedure combines the SET_USER and SET_SESSION_ID procedures to create one call.

Syntax

```
APEX_CUSTOM_AUTH.DEFINE_USER_SESSION(
    p_user         IN    VARCHAR2,
    p_session_id   IN    NUMBER);
```

Parameters

Table 5–2 describes the parameters available in the DEFINE_USER_SESSION procedure.

Table 5–2 DEFINE_USER_SESSION Parameters

Parameter	Description
p_user	Login name of the user.
p_session_id	The session ID.

Example

In the following example, a new session ID is generated and registered along with the current application user.

```
APEX_CUSTOM_AUTH.DEFINE_USER_SESSION (
    :APP_USER,
    APEX_CUSTOM_AUTH.GET_NEXT_SESSION_ID);
```

See Also: "SET_USER Procedure" on page 5-21 and "SET_SESSION_ID Procedure" on page 5-19.

GET_COOKIE_PROPS Procedure

This procedure obtains the properties of the session cookie used in the current authentication scheme for the specified application. These properties can be viewed directly in the Application Builder by viewing the authentication scheme cookie attributes.

Syntax

```
APEX_CUSTOM_AUTH.GET_COOKIE_PROPS(
    p_app_id                       IN  NUMBER,
    p_cookie_name                  OUT VARCHAR2,
    p_cookie_path                  OUT VARCHAR2,
    p_cookie_domain                OUT VARCHAR2
    p_secure                       OUT BOOLEAN);
```

Parameters

Table 5–3 describes the parameters available in the `GET_COOKIE_PROPS` procedure.

Table 5–3 GET_COOKIE_PROPS Parameters

Parameter	Description
`p_app_id`	An application ID in the current workspace.
`p_cookie_name`	The cookie name.
`p_cookie_path`	The cookie path.
`p_cookie_domain`	The cookie domain.
`p_secure`	Flag to set secure property of cookie.

Example

The following example retrieves the session cookie values used by the authentication scheme of the current application.

```
DECLARE
    l_cookie_name   varchar2(256);
    l_cookie_path   varchar2(256);
    l_cookie_domain varchar2(256);
    l_secure        boolean;
BEGIN
    APEX_CUSTOM_AUTH.GET_COOKIE_PROPS(
        p_app_id => 2918,
        p _cookie_name => l_cookie_name,
        p _cookie_path => l_cookie_path,
        p _cookie_domain => l_cookie_domain,
        p_secure => l_secure);
END;
```

GET_LDAP_PROPS Procedure

This procedure obtains the LDAP attributes of the current authentication scheme for the current application. These properties can be viewed directly in Application Builder by viewing the authentication scheme attributes.

Syntax

```
APEX_CUSTOM_AUTH.GET_LDAP_PROPS(
    p_ldap_host               OUT VARCHAR2,
    p_ldap_port               OUT INTEGER,
    p_use_ssl                 OUT VARCHAR2,
    p_use_exact_dn            OUT VARCHAR2,
    p_search_filter           OUT VARCHAR2,
    p_ldap_dn                 OUT VARCHAR2,
    p_ldap_edit_function      OUT VARCHAR2);
```

Parameters

Table 5–4 describes the parameters available in the `GET_LDAP_PROPS` procedure.

Table 5–4 GET_LDAP_PROPS Parameters

Parameter	Description
`p_ldap_host`	LDAP host name.
`p_ldap_port`	LDAP port number.
`p_use_ssl`	Whether or not SSL is used.
`p_use_exact_dn`	Whether or not exact ditinguished names are used.
`p_search_filter`	The search filter used if exact DN is not used.
`p_ldap_dn`	LDAP DN string.
`p_ldap_edit_function`	LDAP edit function name.

Example

The following example retrieves the LDAP attributes associated with the current application.

```
DECLARE
    l_ldap_host          VARCHAR2(256);
    l_ldap_port          INTEGER;
    l_use_ssl            VARCHAR2(1);
    l_use_exact_dn       VARCHAR2(1);
    l_search_filter      VARCHAR2(256);
    l_ldap_dn            VARCHAR2(256);
    l_ldap_edit_function VARCHAR2(256);
BEGIN
APEX_CUSTOM_AUTH.GET_LDAP_PROPS (
    p_ldap_host       => l_ldap_host,
    p_ldap_port       => l_ldap_port,
    p_use_ssl         => l_use_ssl,
    p_use_exact_dn    => l_use_exact_dn,
    p_search_filter   => l_search_filter,
    p_ldap_dn         => l_ldap_dn,
    p_ldap_edit_function => l_ldap_edit_function);
```

```
END;
```

GET_NEXT_SESSION_ID Function

This function generates the next session ID from the Oracle Application Express sequence generator. This function returns a number.

Syntax

```
APEX_CUSTOM_AUTH.GET_NEXT_SESSION_ID
RETURN NUMBER;
```

Example

The following example generates the next session ID and stores it into a variable.

```
DECLARE
    VAL NUMBER;
BEGIN
    VAL := APEX_CUSTOM_AUTH.GET_NEXT_SESSION_ID;
END;
```

GET_SECURITY_GROUP_ID Function

This function returns a number with the value of the security group ID that identifies the workspace of the current user.

Syntax

```
APEX_CUSTOM_AUTH.GET_SECURITY_GROUP_ID
RETURN NUMBER;
```

Example

The following example retrieves the Security Group ID for the current user.

```
DECLARE
    VAL NUMBER;
BEGIN
    VAL := APEX_CUSTOM_AUTH.GET_SECURITY_GROUP_ID;
END;
```

GET_SESSION_ID Function

This function returns `APEX_APPLICATION.G_INSTANCE` global variable. `GET_SESSION_ID` returns a number.

Syntax

```
APEX_CUSTOM_AUTH.GET_SESSION_ID
RETURN NUMBER;
```

Example

The following example retrieves the session ID for the current user.

```
DECLARE
    VAL NUMBER;
BEGIN
    VAL := APEX_CUSTOM_AUTH.GET_SESSION_ID;
END;
```

GET_SESSION_ID_FROM_COOKIE Function

This function returns the Oracle Application Express session ID located by the session cookie in the context of a page request in the current browser session.

Syntax

```
APEX_CUSTOM_AUTH.GET_SESSION_ID_FROM_COOKIE
RETURN NUMBER;
```

Example

The following example retrieves the session ID from the current session cookie.

```
DECLARE
    VAL NUMBER;
BEGIN
    VAL := APEX_CUSTOM_AUTH.GET_SESSION_ID_FROM_COOKIE;
END;
```

GET_USER Function

This function returns the APEX_APPLICATION.G_USER global variable (VARCHAR2).

Syntax

```
APEX_CUSTOM_AUTH.GET_USER
RETURN VARCHAR2;
```

Examples

The following example retrieves the username associated with the current session.

```
DECLARE
    VAL VARCHAR2(256);
BEGIN
    VAL := APEX_CUSTOM_AUTH.GET_USER;
END;
```

GET_USERNAME Function

This function returns user name registered with the current Oracle Application Express session in the internal sessions table. This user name is usually the same as the authenticated user running the current page.

Syntax

```
APEX_CUSTOM_AUTH.GET_USERNAME
RETURN VARCHAR2;
```

Example

The following example retrieves the username registered with the current application session.

```
DECLARE
    VAL VARCHAR2(256);
BEGIN
    VAL := APEX_CUSTOM_AUTH.GET_USERNAME;
END;
```

IS_SESSION_VALID Function

This function is a Boolean result obtained from executing the current application's authentication scheme to determine if a valid session exists. This function returns the Boolean result of the authentication scheme's page sentry.

Syntax

```
APEX_CUSTOM_AUTH.IS_SESSION_VALID
RETURN BOOLEAN;
```

Example

The following example verifies whether the current session is valid.

```
DECLARE
    L_VAL BOOLEAN;
BEGIN
    L_VAL := APEX_CUSTOM_AUTH.IS_SESSION_VALID;
    IF L_VAL THEN
        htp.p('Valid');
    ELSE
        htp.p('Invalid');
    END IF;
END;
```

LOGIN Procedure

Also referred to as the "Login API," this procedure performs authentication and session registration.

Syntax

```
APEX_CUSTOM_AUTH.LOGIN(
    p_uname                     IN  VARCHAR2  DEFAULT NULL,
    p_password                  IN  VARCHAR2  DEFAULT NULL,
    p_session_id                IN  VARCHAR2  DEFAULT NULL,
    p_app_page                  IN  VARCHAR2  DEFAULT NULL,
    p_entry_point               IN  VARCHAR2  DEFAULT NULL,
    p_preserve_case             IN  BOOLEAN   DEFAULT FALSE);
```

Parameter

Table 5–5 describes the parameters available in the `LOGIN` procedure.

Table 5–5 LOGIN Parameters

Parameter	Description
`p_uname`	Login name of the user.
`p_password`	Clear text user password.
`p_session_id`	Current Oracle Application Express session ID.
`p_app_page`	Current application ID. After login page separated by a colon (:).
`p_entry_point`	Internal use only.
`p_preserve_case`	If true, do not upper `p_uname` during session registration

Example

The following example performs the user authentication and session registration.

```
BEGIN
    APEX_CUSTOM_AUTH.LOGIN (
        p_uname       => 'FRANK',
        p_password    => 'secret99',
        p_session_id  => V('APP_SESSION'),
        p_app_page    => :APP_ID||':1');
END;
```

Note: Do not use bind variable notations for `p_session_id` argument.

LOGOUT Procedure

This procedure causes a logout from the current session by unsetting the session cookie and redirecting to a new location.

Syntax

```
APEX_CUSTOM_AUTH.LOGOUT(
    p_this_app                  IN VARCHAR2  DEFAULT NULL,
    p_next_app_page_sess        IN VARCHAR2  DEFAULT NULL,
    p_next_url                  IN VARCHAR2  DEFAULT NULL);
```

Parameter

Table 5–6 describes the parameters available in the `LOGOUT` procedure.

Table 5–6 LOGOUT Parameters

Parameter	Description
`p_this_app`	Current application ID.
`p_next_app_page_sess`	Application and page number to redirect to. Separate multiple pages using a colon (:) and optionally followed by a colon (:) and the session ID (if control over the session ID is desired).
`p_next_url`	URL to redirect to (use this instead of `p_next_app_page_sess`).

Example

The following example causes a logout from the current session and redirects to page `99` of application `1000`.

```
BEGIN
    APEX_CUSTOM_AUTH.LOGOUT (
        p_this_app            => '1000',
        p_next_app_page_sess  => '1000:99');
END;
```

POST_LOGIN Procedure

This procedure performs session registration, assuming the authentication step has been completed. It can be called only from within an Oracle Application Express application page context.

Syntax

```
APEX_CUSTOM_AUTH.POST_LOGIN(
    p_uname                   IN  VARCHAR2  DEFAULT NULL,
    p_session_id              IN  VARCHAR2  DEFAULT NULL,
    p_app_page                IN  VARCHAR2  DEFAULT NULL,
    p_preserve_case           IN  BOOLEAN   DEFAULT FALSE);
```

Parameter

Table 5–7 describes the parameters available in the `POST_LOGIN` procedure.

Table 5–7 POST_LOGIN Parameters

Parameter	Description
`p_uname`	Login name of user.
`p_session_id`	Current Oracle Application Express session ID.
`p_app_page`	Current application ID and after login page separated by a colon (:).
`p_preserve_case`	If true, do not include `p_uname` in uppercase during session registration.

Example

The following example performs the session registration following a successful authentication.

```
BEGIN
    APEX_CUSTOM_AUTH.POST_LOGIN (
        p_uname       => 'FRANK',
        p_session_id  => V('APP_SESSION'),
        p_app_page    => :APP_ID||':1');
END;
```

SESSION_ID_EXISTS Function

This function returns a Boolean result based on the global package variable containing the current Oracle Application Express session ID. Returns true if the result is a positive number and returns false if the result is a negative number.

Syntax

```
APEX_CUSTOM_AUTH.SESSION_ID_EXISTS
RETURN BOOLEAN;
```

Example

The following example checks whether the current session ID is valid and exists.

```
DECLARE
    L_VAL BOOLEAN;
BEGIN
    L_VAL := APEX_CUSTOM_AUTH.SESSION_ID_EXISTS;
    IF VAL THEN
        htp.p('Exists');
    ELSE
        htp.p('Does not exist');
    END IF;
END;
```

SET_SESSION_ID Procedure

This procedure sets APEX_APPLICATION.G_INSTANCE global variable. This procedure requires the parameter P_SESSION_ID (NUMBER) which specifies a session ID.

Syntax

```
APEX_CUSTOM_AUTH.SET_SESSION_ID(
    p_session_id    IN    NUMBER);
```

Parameters

Table 5–8 describes the parameters available in the SET_SESSION_ID procedure.

Table 5–8 SET_SESSION_ID Parameters

Parameter	Description
p_session_id	The session ID to be registered.

Example

In the following example, the session ID value registered is retrieved from the browser cookie.

```
APEX_CUSTOM_AUTH.SET_SESSION_ID(APEX_CUSTOM_AUTH.GET_SESSION_ID_FROM_COOKIE);
```

SET_SESSION_ID_TO_NEXT_VALUE Procedure

This procedure combines the operation of GET_NEXT_SESSION_ID and SET_SESSION_ID in one call.

Syntax

```
APEX_CUSTOM_AUTH.SET_SESSION_ID_TO_NEXT_VALUE;
```

Example

In the following example, if the current session is not valid, a new session ID is generated and registered.

```
IF NOT APEX_CUSTOM_AUTH.SESSION_ID_EXISTS THEN
    APEX_CUSTOM_AUTH.SET_SESSION_ID_TO_NEXT_VALUE;
END IF;
```

SET_USER Procedure

This procedure sets the APEX_APPLICATION.G_USER global variable. SET_USER requires the parameter P_USER (VARCHAR2) which defines a user ID.

Syntax

```
APEX_CUSTOM_AUTH.SET_USER(
    p_user   IN    VARCHAR2);
```

Parameters

Table 5–9 describes the parameters available in the SET_USER procedure.

Table 5–9 SET_USER Parameters

Parameter	Description
p_user	The user ID to be registered.

Example

In the following example, if the current application user is **NOBODY**, then **JOHN.DOE** is registered as the application user.

```
IF V('APP_USER') = 'NOBODY' THEN
    APEX_CUSTOM_AUTH.SET_USER('JOHN.DOE');
END IF;
```

6

APEX_LDAP

You can use `APEX_LDAP` to perform various operations related to Lightweight Directory Access Protocol (LDAP) authentication.

Topics:

- AUTHENTICATE Function
- GET_ALL_USER_ATTRIBUTES Procedure
- GET_USER_ATTRIBUTES Procedure
- IS_MEMBER Function
- MEMBER_OF Function
- MEMBER_OF2 Function

AUTHENTICATE Function

The AUTHENTICATE function returns a boolean true if the user name and password can be used to perform a SIMPLE_BIND_S, call using the provided search base, host, and port.

Syntax

```
APEX_LDAP.AUTHENTICATE(
    p_username     IN VARCHAR2 DEFAULT NULL,
    p_password     IN VARCHAR2 DEFAULT NULL,
    p_search_base  IN VARCHAR2,
    p_host         IN VARCHAR2,
    p_port         IN VARCHAR2 DEFAULT 389,
    p_use_ssl      IN VARCHAR2 DEFAULT 'N')
RETURN BOOLEAN;
```

Parameters

Table 6–1 describes the parameters available in the AUTHENTICATE function.

Table 6–1 AUTHENTICATE Parameters

Parameter	Description
p_username	Login name of the user.
p_password	Password for p_username.
p_search_base	LDAP search base, for example, dc=users,dc=my,dc=org.
p_host	LDAP server host name.
p_port	LDAP server port number.
p_use_ssl	Set to 'Y' to use SSL in bind to LDAP server. Set to 'A' to use SSL with one way authentication (requires LDAP server certificate configured in an Oracle wallet). Set to 'N' to not use SSL (default).

Example

The following example demostrates how to use the APEX_LDAP.AUTHENTICATE function to verify user credentials against an LDAP Server.

```
IF APEX_LDAP.AUTHENTICATE(
    p_username =>'firstname.lastname',
    p_password =>'abcdef',
    p_search_base => 'cn=user,l=amer,dc=my_company,dc=com',
    p_host => 'our_ldap_sever.my_company.com',
    p_port => 389) THEN
    dbms_output.put_line('authenticated');
ELSE
    dbms_output.put_line('authentication failed');
END IF;
```

GET_ALL_USER_ATTRIBUTES Procedure

The GET_ALL_USER_ATTRIBUTES procedure returns two OUT arrays of user_attribute names and values for the user name designated by p_username (with password if required) using the provided auth base, host, and port.

Syntax

```
APEX_LDAP.GET_ALL_USER_ATTRIBUTES(
    p_username          IN VARCHAR2 DEFAULT NULL,
    p_pass              IN VARCHAR2 DEFAULT NULL,
    p_auth_base         IN VARCHAR2 DEFAULT NULL,
    p_host              IN VARCHAR2,
    p_port              IN VARCHAR2 DEFAULT 389,
    p_use_ssl           IN VARCHAR2 DEFAULT 'N',
    p_attributes        OUT wwv_flow_global.vc_arr2,
    p_attribute_values  OUT wwv_flow_global.vc_arr2);
```

Parameters

Table 6–2 describes the parameters for the GET_ALL_USER_ATTRIBUTES procedure.

Table 6–2 GET_ALL_USER_ATTRIBUTES Parameters

Parameter	Description
p_username	Login name of the user.
p_pass	Password for p_username.
p_auth_base	LDAP search base, for example, dc=users,dc=my,dc=org.
p_host	LDAP server host name.
p_port	LDAP server port number.
p_use_ssl	Set to 'Y' to use SSL in bind to LDAP server. Set to 'A' to use SSL with one way authentication (requires LDAP server certificate configured in an Oracle wallet). Set to 'N' to not use SSL (default).
p_attributes	An array of attribute names returned.
p_attribute_values	An array of values returned for each corresponding attribute name returned in p_attributes.

Example

The following example demonstrates how to use the APEX_LDAP.GET_ALL_USER_ATTRIBUTES procedure to retrieve all attribute value's associated to a user.

```
DECLARE
    L_ATTRIBUTES       wwv_flow_global.vc_arr2;
    L_ATTRIBUTE_VALUES wwv_flow_global.vc_arr2;
BEGIN
    APEX_LDAP.GET_ALL_USER_ATTRIBUTES(
        p_username         => 'firstname.lastname',
        p_pass             => 'abcdef',
        p_auth_base        => 'cn=user,l=amer,dc=my_company,dc=com',
        p_host             => 'our_ldap_sever.my_company.com',
        p_port             => '389',
        p_attributes       => L_ATTRIBUTES,
        p_attribute_values => L_ATTRIBUTE_VALUES);
```

```
    FOR i IN L_ATTRIBUTES.FIRST..L_ATTRIBUTES.LAST LOOP
        htp.p('attribute name: '||L_ATTRIBUTES(i));
        htp.p('attribute value: '||L_ATTRIBUTE_VALUES(i));
    END LOOP;
END;
```

GET_USER_ATTRIBUTES Procedure

The GET_USER_ATTRIBUTES procedure returns an OUT array of user_attribute values for the user name designated by p_username (with password if required) corresponding to the attribute names passed in p_attributes using the provided auth base, host, and port.

Syntax

```
APEX_LDAP.GET_USER_ATTRIBUTES(
    p_username          IN VARCHAR2 DEFAULT NULL,
    p_pass              IN VARCHAR2 DEFAULT NULL,
    p_auth_base         IN VARCHAR2,
    p_host              IN VARCHAR2,
    p_port              IN VARCHAR2 DEFAULT 389,
    p_use_ssl           IN VARCHAR2 DEFAULT 'N',
    p_attributes        IN  wwv_flow_global.vc_arr2,
    p_attribute_values  OUT wwv_flow_global.vc_arr2);
```

Parameters

Table 6–3 describes the parameters available in the GET_USER_ATTRIBUTES procedure.

Table 6–3 GET_USER_ATTRIBUTES Parameters

Parameter	Description
p_username	Login name of the user.
p_pass	Password for p_username.
p_auth_base	LDAP search base, for example, dc=users,dc=my,dc=org.
p_host	LDAP server host name.
p_port	LDAP server port number.
p_use_ssl	Set to 'Y' to use SSL in bind to LDAP server. Set to 'A' to use SSL with one way authentication (requires LDAP server certificate configured in an Oracle wallet). Set to 'N' to not use SSL (default).
p_attributes	An array of attribute names for which values are to be returned.
p_attribute_values	An array of values returned for each corresponding attribute name in p_attributes.

Example

The following example demonstrates how to use the APEX_LDAP.GET_USER_ATTRIBUTES procedure to retrieve a specific attribute value associated to a user.

```
DECLARE
    L_ATTRIBUTES wwv_flow_global.vc_arr2;
    L_ATTRIBUTE_VALUES wwv_flow_global.vc_arr2;
BEGIN
    L_ATTRIBUTES(1) := 'xxxxxxxxxx'; /* name of the employee number attribute */
    APEX_LDAP.GET_USER_ATTRIBUTES(
        p_username => 'firstname.lastname',
        p_pass => NULL,
        p_auth_base => 'cn=user,l=amer,dc=my_company,dc=com',
        p_host => 'our_ldap_sever.my_company.com',
        p_port => '389',
```

```
        p_attributes => L_ATTRIBUTES,
        p_attribute_values => L_ATTRIBUTE_VALUES);
END;
```

IS_MEMBER Function

The IS_MEMBER function returns a boolean true if the user named by p_username (with password if required) is a member of the group specified by the p_group and p_group_base parameters using the provided auth base, host, and port.

Syntax

```
APEX_LDAP.IS_MEMBER(
    p_username     IN VARCHAR2,
    p_pass         IN VARCHAR2 DEFAULT NULL,
    p_auth_base    IN VARCHAR2,
    p_host         IN VARCHAR2,
    p_port         IN VARCHAR2 DEFAULT 389,
    p_use_ssl      IN VARCHAR2 DEFAULT 'N',
    p_group        IN VARCHAR2,
    p_group_base   IN VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 6–4 describes the parameters available in the IS_MEMBER function.

Table 6–4 IS_MEMBER Parameters

Parameter	Description
p_username	Login name of the user.
p_pass	Password for p_username.
p_auth_base	LDAP search base, for example, dc=users,dc=my,dc=org.
p_host	LDAP server host name.
p_port	LDAP server port number.
p_use_ssl	Set to 'Y' to use SSL in bind to LDAP server. Set to 'A' to use SSL with one way authentication (requires LDAP server certificate configured in an Oracle wallet). Set to 'N' to not use SSL.
p_group	Name of the group to be search for membership.
p_group_base	The base from which the search should be started.

Example

The following example demonstrates how to use the APEX_LDAP.IS_MEMBER function to verify whether a user is a member of a group against an LDAP server.

```
DECLARE
    L_VAL boolean;
BEGIN
    L_VAL := APEX_LDAP.IS_MEMBER(
        p_username =>'firstname.lastname',
        p_pass =>'abcdef',
        p_auth_base => 'cn=user,l=amer,dc=my_company,dc=com',
        p_host => 'our_ldap_sever.my_company.com',
        p_port => 389,
        p_group => 'group_name',
        p_group_base => 'group_base');
    IF L_VAL THEN
        htp.p('Is a member.');
```

```
    ELSE
        htp.p('Not a member.');
    END IF;
END;
```

MEMBER_OF Function

The MEMBER_OF function returns an array of groups the user name designated by p_username (with password if required) belongs to, using the provided auth base, host, and port.

Syntax

```
APEX_LDAP.MEMBER_OF(
    p_username     IN VARCHAR2 DEFAULT NULL,
    p_pass         IN VARCHAR2 DEFAULT NULL,
    p_auth_base    IN VARCHAR2,
    p_host         IN VARCHAR2,
    p_port         IN VARCHAR2 DEFAULT 389,
    p_use_ssl      IN VARCHAR2 DEFAULT 'N')
RETURN wwv_flow_global.vc_arr2;
```

Parameters

Table 6–5 describes the parameters available in the MEMBER_OF function.

Table 6–5 MEMBER_OF Parameters

Parameter	Description
p_username	Login name of the user.
p_pass	Password for p_username.
p_auth_base	LDAP search base, for example, dc=users,dc=my,dc=org.
p_host	LDAP server host name.
p_port	LDAP server port number.
p_use_ssl	Set to 'Y' to use SSL in bind to LDAP server. Set to 'A' to use SSL with one way authentication (requires LDAP server certificate configured in an Oracle wallet). Set to 'N' to not use SSL (default).

Example

The following example demonstrates how to use the APEX_LDAP.MEMBER_OF function to retrieve all the groups designated by the specified username.

```
DECLARE
    L_MEMBERSHIP       wwv_flow_global.vc_arr2;
BEGIN
    L_MEMBERSHIP := APEX_LDAP.MEMBER_OF(
        p_username         => 'firstname.lastname',
        p_pass             => 'abcdef',
        p_auth_base        => 'cn=user,l=amer,dc=my_company,dc=com',
        p_host             => 'our_ldap_sever.my_company.com',
        p_port             => '389');
    FOR i IN L_MEMBERSHIP.FIRST..L_MEMBERSHIP.LAST LOOP
        htp.p('Member of: '||L_MEMBERSHIP(i));
    END LOOP;
END;
```

MEMBER_OF2 Function

The MEMBER_OF2 function returns a VARCHAR2 colon delimited list of groups the user name designated by p_username (with password if required) belongs to, using the provided auth base, host, and port.

Syntax

```
APEX_LDAP.MEMBER_OF2(
    p_username     IN VARCHAR2 DEFAULT NULL,
    p_pass         IN VARCHAR2 DEFAULT NULL,
    p_auth_base    IN VARCHAR2,
    p_host         IN VARCHAR2,
    p_port         IN VARCHAR2 DEFAULT 389,
    p_use_ssl      IN VARCHAR2 DEFAULT 'N')
RETURN VARCHAR2;
```

Parameters

Table 6–6 describes the parameters available in the MEMBER_OF2 function.

Table 6–6 MEMBER_OF2 Parameters

Parameter	Description
p_username	Login name of the user.
p_pass	Password for p_username.
p_auth_base	LDAP search base, for example, dc=users,dc=my,dc=org.
p_host	LDAP server host name.
p_port	LDAP server port number.
p_use_ssl	Set to 'Y' to use SSL in bind to LDAP server. Set to 'A' to use SSL with one way authentication (requires LDAP server certificate configured in an Oracle wallet). Set to 'N' to not use SSL (default).

Example

The following example demonstrates how to use the APEX_LDAP.MEMBER_OF2 function to retreive all the groups designated by the specified username.

```
DECLARE
    L_VAL varchar2(4000);
BEGIN
    L_VAL := APEX_LDAP.MEMBER_OF2(
        p_username => 'firstname.lastname',
        p_pass => 'abcdef',
        p_auth_base => 'cn=user,l=amer,dc=my_company,dc=com',
        p_host => 'our_ldap_sever.my_company.com',
        p_port => 389);
    htp.p('Is Member of:'||L_VAL);
END;
```

7

APEX_INSTANCE_ADMIN

The APEX_INSTANCE_ADMIN package provides utilities for managing an Oracle Application Express runtime environment. You use the APEX_INSTANCE_ADMIN package to get and set email settings, wallet settings, report printing settings and to manage scheme to workspace mappings. APEX_INSTANCE_ADMIN can be executed by the SYS, SYSTEM, and APEX_030200 database users as well as any database user granted the role APEX_ADMINISTRATOR_ROLE.

Topics:

- ADD_SCHEMA Procedure
- ADD_WORKSPACE Procedure
- GET_PARAMETER Function
- GET_SCHEMAS Function
- REMOVE_SAVED_REPORTS Procedure
- REMOVE_SCHEMA Procedure
- REMOVE_WORKSPACE Procedure
- SET_PARAMETER Procedure
- Available Parameter Values

ADD_SCHEMA Procedure

The ADD_SCHEMA procedure adds a schema to a workspace to schema mapping.

Syntax

```
APEX_INSTANCE_ADMIN.ADD_SCHEMA(
    p_workspace    IN VARCHAR2,
    p_schema       IN VARCHAR2);
```

Parameters

Table 7–1 describes the parameters available in the ADD_SCHEMA procedure.

Table 7–1 ADD_SCHEMA Parameters

Parameter	Description
p_workspace	The name of the workspace to which the schema mapping will be added.
p_schema	The schema to add to the schema to workspace mapping.

Example

The following example demonstrates how to use the ADD_SCHEMA procedure to map a schema mapped to a workspace.

```
BEGIN
    APEX_INSTANCE_ADMIN.ADD_SCHEMA('MY_WORKSPACE','FRANK');
END;
```

ADD_WORKSPACE Procedure

The ADD_WORKSPACE procedure adds a workspace to an Application Express Instance.

Syntax

```
APEX_INSTANCE_ADMIN.ADD_WORKSPACE(
    p_workspace_id        IN NUMBER DEFAULT NULL,
    p_workspace           IN VARCHAR2,
    p_primary_schema      IN VARCHAR2,
    p_additional_schemas  IN VARCHAR2 );
```

Parameters

Table 7–2 describes the parameters available in the ADD_WORKSPACE procedure.

Table 7–2 ADD_WORKSPACE Parameters

Parameter	Description
p_workspace_id	The ID to uniquely identify the workspace in an Application Express instance. This may be left null and a new unique ID will be assigned.
p_workspace	The name of the workspace to be added.
p_primary_schema	The primary database schema to associate with the new workspace.
p_additional_schemas	A colon delimited list of additional schemas to associate with this workspace.

Example

The following example demonstrates how to use the ADD_WORKSPACE procedure to add a new workspace named MY_WORKSPACE using the primary schema, SCOTT, along with additional schema mappings for HR and OE.

```
BEGIN
    APEX_INSTANCE_ADMIN.ADD_WORKSPACE(8675309,'MY_WORKSPACE','SCOTT','HR:OE');
END;
```

GET_PARAMETER Function

The GET_PARAMETER function retrieves the value of a parameter used in administering a runtime environment.

Syntax

```
APEX_INSTANCE_ADMIN.GET_PARAMETER(
    p_parameter     IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 7–3 describes the parameters available in the GET_PARAMETER function.

Table 7–3 GET_PARAMETER Parameters

Parameter	Description
p_parameter	The instance parameter to be retrieved. See "Available Parameter Values" on page 7-11.

Example

The following example demonstrates how to use the GET_PARAMETER function to retrieve the SMTP_HOST_ADDRESS parameter currently defined for an Oracle Application Express instance.

```
DECLARE
    L_VAL VARCHAR2(4000);
BEGIN
    L_VAL :=APEX_INSTANCE_ADMIN.GET_PARAMETER('SMTP_HOST_ADDRESS');
    DBMS_OUTPUT.PUT_LINE('The SMTP Host Setting Is: '||L_VAL);
END;
```

GET_SCHEMAS Function

The GET_SCHEMAS function retrieves a comma-delimited list of schemas that are mapped to a given workspace.

Syntax

```
APEX_INSTANCE_ADMIN.GET_SCHEMAS(
    p_workspace     IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 7–4 describes the parameters available in the GET_SCHEMAS function.

Table 7–4 GET_SCHEMAS Parameters

Parameter	Description
p_workspace	The name of the workspace from which to retrieve the schema list.

Example

The following example demonstrates how to use the GET_SCHEMA function to retrieve the underlying schemas mapped to a workspace.

```
DECLARE
    L_VAL VARCHAR2(4000);
BEGIN
    L_VAL :=APEX_INSTANCE_ADMIN.GET_SCHEMAS('MY_WORKSPACE');
    DBMS_OUTPUT.PUT_LINE('The schemas for my workspace: '||L_VAL);
END;
```

REMOVE_SAVED_REPORT Procedure

The REMOVE_SAVED_REPORT procedure removes a specific user's saved interactive report settings for a particular application.

Syntax

```
APEX_INSTANCE_ADMIN.REMOVE_SAVED_REPORT(
    p_application_id     IN NUMBER,
    p_report_id          IN NUMBER);
```

Parameters

Table 7–5 describes the parameters available in the REMOVE_SAVED_REPORT procedure.

Table 7–5 REMOVE_SAVED_REPORT Parameters

Parameter	Description
p_application_id	The ID of the application for which to remove user saved interactive report information.
p_report_id	The ID of the saved user interactive report to be removed.

Example

The following example demonstrates how to use the REMOVE_SAVED_REPORT procedure to remove user saved interactive report with the ID 123 for the application with an ID of 100.

```
BEGIN
    APEX_INSTANCE_ADMIN.REMOVE_SAVED_REPORT(100,123);
END;
```

REMOVE_SAVED_REPORTS Procedure

The REMOVE_SAVED_REPORTS procedure removes all user saved interactive report settings for a particular application or for the entire instance.

Syntax

```
APEX_INSTANCE_ADMIN.REMOVE_SAVED_REPORTS(
    p_application_id     IN NUMBER DEFAULT NULL);
```

Parameters

Table 7–6 describes the parameters available in the REMOVE_SAVED_REPORTS procedure.

Table 7–6 REMOVE_SAVED_REPORTS Parameters

Parameter	Description
p_application_id	The ID of the application for which to remove user saved interactive report information. If this parameter is left null, all user saved interactive reports for the entire instance will be removed.

Example

The following example demonstrates how to use the REMOVE_SAVED_REPORTS procedure to remove user saved interactive report information for the application with an ID of 100.

```
BEGIN
    APEX_INSTANCE_ADMIN.REMOVE_SAVED_REPORTS(100);
END;
```

REMOVE_SCHEMA Procedure

This REMOVE_SCHEMA procedure removes a workspace to schema mapping.

Syntax

```
APEX_INSTANCE_ADMIN.REMOVE_SCHEMA(
    p_workspace     IN VARCHAR2,
    p_schema        IN VARCHAR2);
```

Parameters

Table 7–7 describes the parameters available in the REMOVE_SCHEMA procedure.

Table 7–7 REMOVE_SCHEMA Parameters

Parameter	Description
p_workspace	The name of the workspace from which the schema mapping will be removed.
p_schema	The schema to remove from the schema to workspace mapping.

Example

The following example demonstrates how to use the REMOVE_SCHEMA procedure to remove the schema named Frank from the MY_WORKSPACE workspace to schema mapping.

```
BEGIN
    APEX_INSTANCE_ADMIN.REMOVE_SCHEMA('MY_WORKSPACE','FRANK');
END;
```

REMOVE_WORKSPACE Procedure

The REMOVE_WORKSPACE procedure removes a workspace from an Application Express instance.

Syntax

```
APEX_INSTANCE_ADMIN.REMOVE_WORKSPACE(
    p_workspace         IN VARCHAR2,
    p_drop_users        IN VARCHAR2 DEFAULT 'N',
    p_drop_tablespaces  IN VARCHAR2 DEFAULT 'N' );
```

Parameters

Table 7–8 describes the parameters available in the REMOVE_WORKSPACE procedure.

Table 7–8 REMOVE_WORKSPACE Parameters

Parameter	Description
p_workspace	The name of the workspace to be removed.
p_drop_users	'Y' to drop the database user associated with the workspace. The default is 'N'.
p_drop_tablespaces	'Y' to drop the tablespace associated with the database user associated with the workspace. The default is 'N'.

Example

The following example demonstrates how to use the REMOVE_WORKSPACE procedure to remove an existing workspace named MY_WORKSPACE, along with the associated database users and tablespace.

```
BEGIN
    APEX_INSTANCE_ADMIN.REMOVE_WORKSPACE('MY_WORKSPACE','Y','Y');
END;
```

SET_PARAMETER Procedure

The SET_PARAMETER procedure sets a parameter used in administering a runtime environment.

Syntax

```
APEX_INSTANCE_ADMIN.SET_PARAMETER(
    p_parameter     IN VARCHAR2,
    p_value         IN VARCHAR2 DEFAULT 'N');
```

Parameters

Table 7–9 describes the parameters available in the SET_PARAMETER procedure.

Table 7–9 SET_PARAMETER Parameters

Parameter	Description
p_parameter	The instance parameter to be set.
p_value	The value of the parameter. See "Available Parameter Values" on page 7-11.

Example

The following example demonstrates how to use the SET_PARAMETER procedure to set the SMTP_HOST_ADDRESS parameter for an Oracle Application Express instance.

```
BEGIN
    APEX_INSTANCE_ADMIN.SET_PARAMETER('SMTP_HOST_ADDRESS','mail.mycompany.com');
END;
```

Available Parameter Values

Table 7–10 lists all the available parameter values you can set within the APEX_INSTANCE_ADMIN package, including parameters for email, wallet, and reporting printing.

Table 7–10 Available Parameters

Parameter Name	Description
SMTP_FROM	Defines the "from" address for administrative tasks that generate email, such as approving a provision request or resetting a password. Enter a valid email address, for example: someone@somewhere.com
SMTP_HOST_ADDRESS	Defines the server address of the SMTP server. If you are using another server as an SMTP relay, change this parameter to that server's address. Default setting: localhost
SMTP_HOST_PORT	Defines the port the SMTP server listens to for mail requests. Default setting: 25
WALLET_PATH	The path to the wallet on the file system, for example: file:/home/<*username*>/wallets
WALLET_PWD	The password associated with the wallet.
PASSWORD_HISTORY_DAYS	Defines the maximum number of days a developer or administrator account password may be used before the account expires. The default value is 45 days.
PRINT_BIB_LICENSED	Specify either standard support or advanced support. Advanced support requires an Oracle BI Publisher license. Valid values include: ■ STANDARD ■ ADVANCED
PRINT_SVR_PROTOCOL	Valid values include: ■ http ■ https
PRINT_SVR_HOST	Specifies the host address of the print server converting engine, for example, localhost. Enter the appropriate host address if the print server is installed at another location.
PRINT_SVR_PORT	Defines the port of the print server engine, for example 8888. Value must be a positive integer.
PRINT_SVR_SCRIPT	Defines the script that is the print server engine, for example: /xmlpserver/convert

Table 7–10 (Cont.) Available Parameters

Parameter Name	Description
REQUIRE_HTTPS	Set to 'Y' to allow authentication pages within the Application Express development and administration applications to be used only when the protocol is HTTPS. Select 'N' to allow these application pages to be used when the protocol is either HTTP or HTTPS.

See Also: "Configuring Email in a Runtime Environment", "Configuring a Wallet in a Runtime Environment", "Configuring Report Printing Settings in a Runtime Environment" in *Oracle Application Express Administration Guide*.

8

APEX_UI_DEFAULT_UPDATE

The `APEX_UI_DEFAULT_UPDATE` package provides procedures to access user interface defaults from within SQL Developer or SQL*Plus.

You can use this package to set the user interface defaults associated with a table within a schema. The package must be called from within the schema that owns the table you are updating.

User interface defaults enable you to assign default user interface properties to a table, column, or view within a specified schema. When you create a form or report using a wizard, the wizard uses this information to create default values for region and item properties. Utilizing user interface defaults can save valuable development time and has the added benefit of providing consistency across multiple pages in an application.

Topics:

- ADD_AD_COLUMN Procedure
- ADD_AD_SYNONYM Procedure
- DEL_AD_COLUMN Procedure
- DEL_AD_SYNONYM Procedure
- DEL_COLUMN Procedure
- DEL_TABLE Procedure
- DEL_TABLE Procedure
- SYNCH_TABLE Procedure
- UPD_AD_COLUMN Procedure
- UPD_AD_SYNONYM Procedure
- UPD_COLUMN Procedure
- UPD_DISPLAY_IN_FORM Procedure
- UPD_DISPLAY_IN_REPORT Procedure
- UPD_FORM_REGION_TITLE Procedure
- UPD_GROUP Procedure
- UPD_ITEM_DISPLAY_HEIGHT Procedure
- UPD_ITEM_DISPLAY_WIDTH Procedure
- UPD_ITEM_FORMAT_MASK Procedure
- UPD_ITEM_HELP Procedure

- UPD_LABEL Procedure
- UPD_REPORT_ALIGNMENT Procedure
- UPD_REPORT_FORMAT_MASK Procedure
- UPD_REPORT_REGION_TITLE Procedure
- UPD_TABLE Procedure

See Also: "Managing User Interface Defaults" in *Oracle Application Express SQL Workshop Guide*

ADD_AD_COLUMN Procedure

Adds a User Interface Default Attribute Dictionary entry with the provided definition. Up to three synonyms can be provided during the creation. Additional synonyms can be added post-creation using apex_ui_default_update.add_ad_synonym. Synonyms share the column definition of their base column.

Syntax

```
APEX_UI_DEFAULT_UPDATE.ADD_AD_COLUMN (
    p_column_name           IN  VARCHAR2,
    p_label                 IN  VARCHAR2  DEFAULT NULL,
    p_help_text             IN  VARCHAR2  DEFAULT NULL,
    p_format_mask           IN  VARCHAR2  DEFAULT NULL,
    p_default_value         IN  VARCHAR2  DEFAULT NULL,
    p_form_format_mask      IN  VARCHAR2  DEFAULT NULL,
    p_form_display_width    IN  VARCHAR2  DEFAULT NULL,
    p_form_display_height   IN  VARCHAR2  DEFAULT NULL,
    p_form_data_type        IN  VARCHAR2  DEFAULT NULL,
    p_report_format_mask    IN  VARCHAR2  DEFAULT NULL,
    p_report_col_alignment  IN  VARCHAR2  DEFAULT NULL,
    p_syn_name1             IN  VARCHAR2  DEFAULT NULL,
    p_syn_name2             IN  VARCHAR2  DEFAULT NULL,
    p_syn_name3             IN  VARCHAR2  DEFAULT NULL);
```

Parameters

Table 8–5 describes the parameters available in the `ADD_AD_COLUMN` procedure.

Table 8–1 ADD_AD_COLUMN Parameters

Parameter	Description
`p_column_name`	Name of column to be created.
`p_label`	Used for item label and report column heading.
`p_help_text`	Used for help text for items and interactive report columns
`p_format_mask`	Used as the format mask for items and report columns. Can be overwritten by report for form specific format masks.
`p_default_value`	Used as the default value for items.
`p_form_format_mask`	If provided, used as the format mask for items, overriding any value for the general format mask.
`p_form_display_width`	Used as the width of any items using this Attribute Definition.
`p_form_display_height`	Used as the height of any items using this Attribute Definition (only used by item types such as text areas and shuttles).
`p_form_data_type`	Used as the data type for items (results in an automatic validation). Valid values are VARCHAR, NUMBER and DATE.
`p_report_format_mask`	Used as the data type for items (results in an automatic validation). Valid values are VARCHAR, NUMBER and DATE.

Table 8–1 (Cont.) ADD_AD_COLUMN Parameters

Parameter	Description
p_report_col_alignment	Used as the data type for items (results in an automatic validation). Valid values are VARCHAR, NUMBER and DATE.
p_syn_name1	Name of synonym to be created along with this column. For more than 3, use APEX_UI_DEFAULT_UPDATE.ADD_AD_SYNONYM.
p_syn_name2	Name of second synonym to be created along with this column. For more than 3, use APEX_UI_DEFAULT_UPDATE.ADD_AD_SYNONYM.
p_syn_name3	Name of third synonym to be created along with this column. For more than 3, use APEX_UI_DEFAULT_UPDATE.ADD_AD_SYNONYM.

Example

The following example creates a new attribute to the UI Defaults Attribute Dictionary within the workspace associated with the current schema. It also creates a synonym for that attribute.

```
BEGIN
    apex_ui_default_update.add_ad_column (
       p_column_name          => 'CREATED_BY',
       p_label                => 'Created By',
       p_help_text            => 'User that created the record.',
       p_form_display_width   => 30,
       p_form_data_type       => 'VARCHAR',
       p_report_col_alignment => 'LEFT',
       p_syn_name1            => 'CREATED_BY_USER' );
END;
```

ADD_AD_SYNONYM Procedure

If the column name is found within the User Interface Default Attribute Dictionary, the synonym provided is created and associated with that column. Synonyms share the column definition of their base column.

Syntax

```
APEX_UI_DEFAULT_UPDATE.ADD_AD_SYNONYM (
    p_column_name           IN VARCHAR2,
    p_syn_name              IN VARCHAR2);
```

Parameters

Table 8–2 describes the parameters available in the ADD_AD_SYNONYM procedure.

Table 8–2 ADD_AD_SYNONYM Parameters

Parameter	Description
p_column_name	Name of column with the Attribute Dictionary that the synonym is being created for.
p_syn_name	Name of synonym to be created.

Example

The following example add the synonym CREATED_BY_USER to the CREATED_BY attribute of the UI Defaults Attribute Dictionary within the workspace associated with the current schema.

```
BEGIN
    apex_ui_default_update.add_ad_synonym (
       p_column_name => 'CREATED_BY',
       p_syn_name    => 'CREATED_BY_USER' );
END;
```

DEL_AD_COLUMN Procedure

If the column name is found within the User Interface Default Attribute Dictionary, the column, along with any associated synonyms, is deleted.

Syntax

```
APEX_UI_DEFAULT_UPDATE.DEL_AD_COLUMN (
    p_column_name           IN VARCHAR2);
```

Parameters

Table 8–3 describes the parameters available in the DEL_AD_COLUMN procedure.

Table 8–3 DEL_AD_COLUMN Parameters

Parameter	Description
p_column_name	Name of column to be deleted

Example

The following example deletes the attribute CREATED_BY from the UI Defaults Attribute Dictionary within the workspace associated with the current schema.

```
BEGIN
    apex_ui_default_update.del_ad_column (
       p_column_name => 'CREATED_BY' );
END;
```

DEL_AD_SYNONYM Procedure

If the synonym name is found within the User Interface Default Attribute Dictionary, the synonym name is deleted.

Syntax

```
APEX_UI_DEFAULT_UPDATE.DEL_AD_SYNONYM (
    p_syn_name           IN VARCHAR2);
```

Parameters

Table 8–4 describes the parameters available in the DEL_AD_SYNONYM procedure.

Table 8–4 DEL_AD_SYNONYM Parameters

Parameter	Description
p_syn_name	Name of synonym to be deleted

Example

The following example deletes the synonym CREATED_BY_USER from the UI Defaults Attribute Dictionary within the workspace associated with the current schema.

```
BEGIN
    apex_ui_default_update.del_ad_synonym (
       p_syn_name    => 'CREATED_BY_USER' );
END;
```

DEL_COLUMN Procedure

If the provided table and column exists within the user's schema's table based User Interface Defaults, the UI Defaults for it is deleted. If a column exists in the base table but does not have column details in the table definition within the UI Defaults Table Dictionary, that column will then be checked against the UI Defaults Attribute Dictionary.

Syntax

```
APEX_UI_DEFAULT_UPDATE.DEL_COLUMN (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2);
```

Parameters

Table 8–5 describes the parameters available in the DEL_COLUMN procedure.

Table 8–5 DEL_COLUMN Parameters

Parameter	Description
p_table_name	Name of table whose column's UI Defaults are to be deleted.
p_column_name	Name of columns whose UI Defaults are to be deleted.

Example

The following example deletes the column CREATED_BY from the EMP table definition within the UI Defaults Table Dictionary within the current schema.

```
BEGIN
    apex_ui_default_update.del_ad_column (
       p_table_name  => 'EMP',
       p_column_name => 'CREATED_BY' );
END;
```

DEL_GROUP Procedure

If the provided table and group exists within the user's schema's table based User Interface Defaults, the UI Defaults for it are deleted and any column within the table that references that group has the group_id set to null.

Syntax

```
APEX_UI_DEFAULT_UPDATE.DEL_GROUP (
    p_table_name            IN VARCHAR2,
    p_group_name            IN VARCHAR2);
```

Parameters

Table 8–6 describes the parameters available in the DEL_GROUP procedure.

Table 8–6 DEL_GROUP Parameters

Parameter	Description
p_table_name	Name of table whose group UI Defaults are to be deleted
p_group_name	Name of group whose UI Defaults are to be deleted

Example

The following example deletes the group AUDIT_INFO from the EMP table definition within the UI Defaults Table Dictionary within the current schema.

```
BEGIN
    apex_ui_default_update.del_group (
       p_table_name => 'EMP',
       p_group_name => 'AUDIT_INFO' );
END;
```

DEL_TABLE Procedure

If the provided table exists within the user's schema's table based User Interface Defaults, the UI Defaults for it is deleted. This includes the deletion of any groups defined for the table and all the columns associated with the table.

Syntax

```
APEX_UI_DEFAULT_UPDATE.DEL_TABLE (
    p_table_name            IN VARCHAR2);
```

Parameters

Table 8–7 describes the parameters available in the `DEL_TABLE` procedure.

Table 8–7 DEL_TABLE Parameters

Parameter	Description
`p_table_name`	Table name

SYNCH_TABLE Procedure

If the Table Based User Interface Defaults for the table do not already exist within the user's schema, they are defaulted. If they do exist, they are synchronized, meaning, the columns in the table will be matched against the column in the UI Defaults Table Definitions. Additions and deletions are used to make them match.

Syntax

```
APEX_UI_DEFAULT_UPDATE.SYNCH_TABLE (
    p_table_name            IN VARCHAR2);
```

Parameters

Table 8–8 describes the parameters available in the SYNCH_TABLE procedure.

Table 8–8 SYNCH_TABLE Parameters

Parameter	Description
p_table_name	Table name

Example

The following example synchronizes the UI Defaults for the EMP table that are associated with the current schema.

```
BEGIN
    apex_ui_default_update.synch_table (
       p_table_name => 'EMP' );
END;
```

UPD_AD_COLUMN Procedure

If the column name is found within the User Interface Default Attribute Dictionary, the column entry is updated using the provided parameters. If 'null%' is passed in, the value of the associated parameter is set to null.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_AD_COLUMN (
    p_column_name           IN  VARCHAR2,
    p_new_column_name       IN  VARCHAR2  DEFAULT NULL,
    p_label                 IN  VARCHAR2  DEFAULT NULL,
    p_help_text             IN  VARCHAR2  DEFAULT NULL,
    p_format_mask           IN  VARCHAR2  DEFAULT NULL,
    p_default_value         IN  VARCHAR2  DEFAULT NULL,
    p_form_format_mask      IN  VARCHAR2  DEFAULT NULL,
    p_form_display_width    IN  VARCHAR2  DEFAULT NULL,
    p_form_display_height   IN  VARCHAR2  DEFAULT NULL,
    p_form_data_type        IN  VARCHAR2  DEFAULT NULL,
    p_report_format_mask    IN  VARCHAR2  DEFAULT NULL,
    p_report_col_alignment  IN  VARCHAR2  DEFAULT NULL);
```

Parameters

Table 8–9 describes the parameters available in the UPD_AD_COLUMN procedure.

Table 8–9 UPD_AD_COLUMN Parameters

Parameter	Description
p_column_name	Name of column to be updated
p_new_column_name	New name for column, if column is being renamed
p_label	Used for item label and report column heading
p_help_text	Used for help text for items and interactive report columns
p_format_mask	Used as the format mask for items and report columns. Can be overwritten by report for form specific format masks.
p_default_value	Used as the default value for items.
p_form_format_mask	If provided, used as the format mask for items, overriding any value for the general format mask.
p_form_display_width	Used as the width of any items using this Attribute Definition.
p_form_display_height	Used as the height of any items using this Attribute Definition (only used by item types such as text areas and shuttles).
p_form_data_type	Used as the data type for items (results in an automatic validation). Valid values are VARCHAR, NUMBER and DATE.
p_report_format_mask	If provided, used as the format mask for report columns, overriding any value for the general format mask.

Table 8–9 (Cont.) UPD_AD_COLUMN Parameters

Parameter	Description
p_report_col_alignment	Used as the alignment for report column data (e.g. number are usually right justified). Valid values are LEFT, CENTER, and RIGHT. Note: If p_label through p_report_col_alignment are set to 'null%', the value will be nullified. If no value is passed in, that column will not be updated.

Example

The following example updates the CREATED_BY column in the UI Defaults Attribute Dictionary within the workspace associated with the current schema, setting the form_format_mask to null.

```
BEGIN
    apex_ui_default_update.upd_ad_column (
       p_column_name      => 'CREATED_BY',
       p_form_format_mask => 'null%');
END;
```

UPD_AD_SYNONYM Procedure

If the synonym name is found within the User Interface Default Attribute Dictionary, the synonym name is updated.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_AD_SYNONYM (
    p_syn_name           IN VARCHAR2,
    p_new_syn_name       IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 8–10 describes the parameters available in the `UPD_AD_SYNONYM` procedure.

Table 8–10 UPD_AD_SYNONYM Parameters

Parameter	Description
`p_syn_name`	Name of synonym to be updated
`p_new_syn_name`	New name for synonym

Example

The following example updates the `CREATED_BY_USER` synonym in the UI Defaults Attribute Dictionary within the workspace associated with the current schema.

```
BEGIN
    apex_ui_default_update.upd_ad_column (
       p_syn_name     => 'CREATED_BY_USER',
       p_new_syn_name => 'USER_CREATED_BY');
END;
```

UPD_COLUMN Procedure

If the provided table and column exists within the user's schema's table based User Interface Defaults, the provided parameters are updated. If 'null%' is passed in, the value of the associated parameter is set to null.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_COLUMN (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_group_id              IN VARCHAR2  DEFAULT NULL,
    p_label                 IN VARCHAR2  DEFAULT NULL,
    p_help_text             IN VARCHAR2  DEFAULT NULL,
    p_display_in_form       IN VARCHAR2  DEFAULT NULL,
    p_display_seq_form      IN VARCHAR2  DEFAULT NULL,
    p_mask_form             IN VARCHAR2  DEFAULT NULL,
    p_default_value         IN VARCHAR2  DEFAULT NULL,
    p_required              IN VARCHAR2  DEFAULT NULL,
    p_display_width         IN VARCHAR2  DEFAULT NULL,
    p_max_width             IN VARCHAR2  DEFAULT NULL,
    p_height                IN VARCHAR2  DEFAULT NULL,
    p_display_in_report     IN VARCHAR2  DEFAULT NULL,
    p_display_seq_report    IN VARCHAR2  DEFAULT NULL,
    p_mask_report           IN VARCHAR2  DEFAULT NULL,
    p_alignment             IN VARCHAR2  DEFAULT NULL);
```

Parameters

Table 8–11 describes the parameters available in the `UPD_COLUMN` procedure.

Table 8–11 UPD_COLUMN Parameters

Parameter	Description
`p_table_name`	Name of table whose column's UI Defaults are being updated
`p_column_name`	Name of column whose UI Defaults are being updated
`p_group_id`	id of group to be associated with the column
`p_label`	When creating a form against this table or view, this will be used as the label for the item if this column is included. When creating a report or tabular form, this will be used as the column heading if this column is included.
`p_help_text`	When creating a form against this table or view, this becomes the help text for the resulting item.
`p_display_in_form`	When creating a form against this table or view, this determines whether this column will be displayed in the resulting form page. Valid values are Y and N.
`p_display_seq_form`	When creating a form against this table or view, this determines the sequence in which the columns will be displayed in the resulting form page.
`p_mask_form`	When creating a form against this table or view, this specifies the mask that will be applied to the item, such as 999-99-9999. This is not used for character based items.
`p_default_value`	When creating a form against this table or view, this specifies the default value for the item resulting from this column.

Table 8–11 (Cont.) UPD_COLUMN Parameters

Parameter	Description
p_required	When creating a form against this table or view, this specifies to generate a validation in which the resulting item must be NOT NULL. Valid values are Y and N.
p_display_width	When creating a form against this table or view, this specifies the display width of the item resulting from this column.
p_max_width	When creating a form against this table or view, this specifies the maximum string length that a user is allowed to enter in the item resulting from this column.
p_height	When creating a form against this table or view, this specifies the display height of the item resulting from this column.
p_display_in_report	When creating a report against this table or view, this determines whether this column will be displayed in the resulting report. Valid values are Y and N.
p_display_seq_report	When creating a report against this table or view, this determines the sequence in which the columns will be displayed in the resulting report.
p_mask_report	When creating a report against this table or view, this specifies the mask that will be applied against the data, such as 999-99-9999. This is not used for character based items.
p_alignment	When creating a report against this table or view, this determines the alignment for the resulting report column. Valid values are L for Left, C for Center, and R for Right. Note: If p_group_id through p_alignment are set to 'null%', the value will be nullified. If no value is passed in, that column will not be updated.

Example

The following example updates the column DEPT_NO within the EMP table definition within the UI Defaults Table Dictionary within the current schema, setting the group_id to null.

```
BEGIN
    apex_ui_default_update.upd_column (
       p_table_name    => 'EMP',
       p_column_name   => 'DEPT_NO',
       p_group_id      => 'null%' );
END;
```

UPD_DISPLAY_IN_FORM Procedure

The UPD_DISPLAY_IN_FORM procedure sets the display in form user interface defaults. This user interface default will be used by wizards when you select to create a form based upon the table. It controls whether the column is included by default or not.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_DISPLAY_IN_FORM (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_display_in_form       IN VARCHAR2);
```

Parameters

Table 8–12 describes the parameters available in the UPD_DISPLAY_IN_FORM procedure.

Table 8–12 UPD_DISPLAY_IN_FORM Parameters

Parameter	Description
p_table_name	Table name
p_column_name	Column name
p_display_in_form	Determines whether or not to display in the form by default, valid values are Y and N

Example

In the following example, when creating a Form against the DEPT table, the display option on the DEPTNO column defaults to 'No'.

```
APEX_UI_DEFAULT_UPDATE.UPD_DISPLAY_IN_FORM(
    p_table_name => 'DEPT',
    p_column_name => 'DEPTNO',
    p_display_in_form => 'N');
```

UPD_DISPLAY_IN_REPORT Procedure

The UPD_DISPLAY_IN_REPORT procedure sets the display in report user interface default. This user interface default is used by wizards when you select to create a report based upon the table and controls whether the column is included by default or not.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_DISPLAY_IN_REPORT (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_display_in_report     IN VARCHAR2);
```

Parameters

Table 8–13 describes the parameters available in the UPD_DISPLAY_IN_REPORT procedure.

Table 8–13 UPD_DISPLAY_IN_REPORT Parameters

Parameter	Description
p_table_name	Table name
p_column_name	Column name
p_display_in_report	Determines whether or not to display in the report by default, valid values are Y and N

Example

In the following example, when creating a Report against the DEPT table, the display option on the DEPTNO column defaults to 'No'.

```
APEX_UI_DEFAULT_UPDATE.UPD_DISPLAY_IN_REPORT(
    p_table_name => 'DEPT',
    p_column_name => 'DEPTNO',
    p_display_in_report => 'N');
```

UPD_FORM_REGION_TITLE Procedure

The UPD_FORM_REGION_TITLE procedure updates the Form Region Title user interface default. User interface defaults are used in wizards when you create a form based upon the specified table.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_FORM_REGION_TITLE (
    p_table_name            IN VARCHAR2,
    p_form_region_title     IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 8–14 describes the parameters available in the UPD_FORM_REGION_TITLE procedure.

Table 8–14 APEX_UI_DEFAULT_UPDATE Parameters

Parameter	Description
p_table_name	Table name
p_form_region_title	Desired form region title

Example

This example demonstrates how to set the Forms Region Title user interface default on the DEPT table.

```
APEX_UI_DEFAULT_UPDATE.UPD_FORM_REGION_TITLE (
    p_table_name         => 'DEPT',
    p_form_region_title  => 'Deptartment Details');
```

UPD_GROUP Procedure

If the provided table and group exist within the user's schema's table based User Interface Defaults, the group name, description and display sequence of the group are updated. If 'null%' is passed in for p_description or p_display_sequence, the value is set to null.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_GROUP (
    p_table_name            IN VARCHAR2,
    p_group_name            IN VARCHAR2,
    p_new_group_name        IN VARCHAR2 DEFAULT NULL,
    p_description           IN VARCHAR2 DEFAULT NULL,
    p_display_sequence      IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 8–15 describes the parameters available in the UPD_GROUP procedure.

Table 8–15 UPD_GROUP Parameters

Parameter	Description
p_table_name	Name of table whose group is being updated
p_group_name	Group being updated
p_new_group_name	New name for group, if group is being renamed
p_description	Description of group
p_display_sequence	Display sequence of group. Note: If p_description or p_display_sequence are set to 'null%', the value will be nullified. If no value is passed in, that column will not be updated.

Example

The following example updates the description of the group AUDIT_INFO within the EMP table definition within the UI Defaults Table Dictionary within the current schema.

```
BEGIN
    apex_ui_default_update.upd_group (
       p_table_name  => 'EMP',
       p_group_name  => 'AUDIT_INFO',
       p_description => 'Audit columns' );
END;
```

UPD_ITEM_DISPLAY_HEIGHT Procedure

The UPD_ITEM_DISPLAY_HEIGHT procedure sets the item display height user interface default. This user interface default is used by wizards when you select to create a form based upon the table and include the specified column. Display height controls if the item is a text box or a text area.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_DISPLAY_HEIGHT (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_display_height        IN NUMBER);
```

Parameters

Table 8–16 describes the parameters available in the UPD_ITEM_DISPLAY_HEIGHT procedure.

Table 8–16 UPD_ITEM_DISPLAY_HEIGHT Parameters

Parameter	Description
p_table_name	Table name
p_column_name	Column name
p_display_height	Display height of any items created based upon this column

Example

The following example sets a default item height of 3 when creating an item on the DNAME column against the DEPT table.

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_DISPLAY_HEIGHT(
   p_table_name => 'DEPT',
   p_column_name => 'DNAME',
   p_display_height => 3);
```

UPD_ITEM_DISPLAY_WIDTH Procedure

The UPD_ITEM_DISPLAY_WIDTH procedure sets the item display width user interface default. This user interface default is used by wizards when you select to create a form based upon the table and include the specified column.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_DISPLAY_WIDTH (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_display_width         IN NUMBER);
```

Parameters

Table 8–17 describes the parameters available in the UPD_ITEM_DISPLAY_WIDTH procedure.

Table 8–17 UPD_ITEM_DISPLAY_WIDTH Parameters

Parameter	Description
p_table_name	Table name
p_column_name	Column name
p_display_width	Display width of any items created based upon this column

Example

The following example sets a default item width of 5 when creating an item on the DEPTNO column against the DEPT table.

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_DISPLAY_WIDTH(
   p_table_name => 'DEPT',
   p_column_name => 'DEPTNO',
   p_display_width => 5);
```

UPD_ITEM_FORMAT_MASK Procedure

The UPD_ITEM_FORMAT_MASK procedure sets the item format mask user interface default. This user interface default is used by wizards when you select to create a form based upon the table and include the specified column. Item format mask is typically used to format numbers and dates.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_FORMAT_MASK (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_format_mask           IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 8–18 describes the parameters available in the UPD_ITEM_FORMAT_MASK procedure.

Table 8–18 UPD_ITEM_FORMAT_MASK Parameters

Parameter	Description
p_table_name	Table name
p_column_name	Column name
p_format_mask	Format mask to be associated with the column

Example

In the following example, when creating a Form against the EMP table, the default item format mask on the HIREDATE column is set to 'DD-MON-YYYY'.

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_FORMAT_MASK(
    p_table_name => 'EMP',
    p_column_name => 'HIREDATE',
    p_format_mask=> 'DD-MON-YYYY');
```

UPD_ITEM_HELP Procedure

The UPD_ITEM_HELP procedure updates the help text for the specified table and column. This user interface default is used when you create a form based upon the table and select to include the specified column.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_HELP (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_help_text             IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 8–19 describes the parameters available in the UPD_ITEM_HELP procedure.

Table 8–19 UPD_ITEM_HELP Parameters

Parameter	Description
p_table_name	Table name
p_column_name	Column name
p_help_text	Desired help text

Example

This example demonstrates how to set the User Interface Item Help Text default for the DEPTNO column in the DEPT table.

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_HELP(
   p_table_name => 'DEPT',
   p_column_name => 'DEPTNO',
   p_help_text => 'The number assigned to the department.');
```

UPD_LABEL Procedure

The UPD_LABEL procedure sets the label used for items. This user interface default is used when you create a form or report based on the specified table and include a specific column.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_ITEM_LABEL (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_label                 IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 8–20 describes the parameters available in the UPD_LABEL procedure.

Table 8–20 UPD__LABEL Parameters

Parameter	Description
p_table_name	Table name
p_column_name	Column name
p_label	Desired item label

Example

This example demonstrates how to set the User Interface Item Label default for the DEPTNO column in the DEPT table.

```
APEX_UI_DEFAULT_UPDATE.UPD_LABEL(
   p_table_name => 'DEPT',
   p_column_name => 'DEPTNO',
   p_label => 'Department Number');
```

UPD_REPORT_ALIGNMENT Procedure

The UPD_REPORT_ALIGNMENT procedure sets the report alignment user interface default. This user interface default is used by wizards when you select to create a report based upon the table and include the specified column and determines if the report column should be left, center, or right justified.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_REPORT_ALIGNMENT (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_report_alignment      IN VARCHAR2);
```

Parameters

Table 8–21 describes the parameters available in the `UPD_REPORT_ALIGNMENT` procedure.

Table 8–21 UPD_REPORT_ALIGNMENT Parameters

Parameter	Description
`p_table_name`	Table name.
`p_column_name`	Column name.
`p_report_alignment`	Defines the alignment of the column in a report. Valid values are L (left), C (center) and R (right).

Example

In the following example, when creating a Report against the DEPT table, the default column alignment on the DEPTNO column is set to Right justified.

```
APEX_UI_DEFAULT_UPDATE.UPD_REPORT_ALIGNMENT(
    p_table_name => 'DEPT',
    p_column_name => 'DEPTNO',
    p_report_alignment => 'R');
```

UPD_REPORT_FORMAT_MASK Procedure

The UPD_REPORT_FORMAT_MASK procedure sets the report format mask user interface default. This user interface default is used by wizards when you select to create a report based upon the table and include the specified column. Report format mask is typically used to format numbers and dates.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_REPORT_FORMAT_MASK (
    p_table_name            IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_format_mask           IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 8–22 describes the parameters available in the UPD_REPORT_FORMAT_MASK procedure.

Table 8–22 UPD_REPORT_FORMAT_MASK Parameters

Parameter	Description
p_table_name	Table name
p_column_name	Column name
p_format_mask	Format mask to be associated with the column whenever it is included in a report

Example

In the following example, when creating a Report against the EMP table, the default format mask on the HIREDATE column is set to 'DD-MON-YYYY'.

```
APEX_UI_DEFAULT_UPDATE.UPD_REPORT_FORMAT_MASK(
    p_table_name => 'EMP',
    p_column_name => 'HIREDATE',
    p_format_mask=> 'DD-MON-YYYY');
```

UPD_REPORT_REGION_TITLE Procedure

The UPD_REPORT_REGION_TITLE procedure sets the Report Region Title. User interface defaults are used in wizards when a report is created on a table.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_REPORT_REGION_TITLE (
    p_table_name            IN VARCHAR2,
    p_report_region_title   IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 8–23 describes the parameters available in the UPD_REPORT_REGION_TITLE procedure.

Table 8–23 UPD_REPORT_REGION_TITLE Parameters

Parameter	Description
p_table_name	Table name
p_report_region_title	Desired report region title

Example

This example demonstrates how to set the Reports Region Title user interface default on the DEPT table.

```
APEX_UI_DEFAULT_UPDATE.UPD_REPORT_REGION_TITLE (
    p_table_name            => 'DEPT',
    p_report_region_title   => 'Deptartments');
```

UPD_TABLE Procedure

If the provided table exists within the user's schema's table based User Interface Defaults, the form region title and report region title are updated to match those provided. If 'null%' is passed in for p_form_region_title or p_report_region_title, the value is set to null.

Syntax

```
APEX_UI_DEFAULT_UPDATE.UPD_TABLE (
    p_table_name            IN VARCHAR2,
    p_form_region_title     IN VARCHAR2 DEFAULT NULL,
    p_report_region_title   IN VARCHAR2 DEFAULT NULL);
```

9

JavaScript APIs

This section describes JavaScript functions and objects included with Oracle Application Express and available on every page. You can use these functions and objects to provide client-side functionality, such as showing and hiding page elements, or making XML HTTP Asynchronous JavaScript and XML (AJAX) requests.

Topics:

- apex.event.trigger(pSelector,pEvent,pData)
- apex.widget.initPageItem(pName,pOptions)
- $x(pNd)
- $v2(pNd)
- $v2(pNd)
- $s(pNd, pValue)
- $u_Carray(pNd)
- $u_Narray(pNd)
- $nvl(pTest, pDefault)
- apex.submit(pRequest | pOptions)
- apex.confirm(pMessage, pRequest)
- $x_Style(pNd, pStyle, pString)
- $x_Hide(pNd)
- $x_Show(pNd)
- $x_Toggle(pNd)
- $x_Remove(pNd)
- $x_Value(pNd,pValue)
- $x_UpTill(pNd, pToTag)
- $x_ItemRow(pNd,pFunc)
- $x_HideItemRow(pNd)
- $x_ShowItemRow(pNd)
- $x_ToggleItemRow(pNd)
- $x_HideAllExcept(pNd,pNdArray)
- $x_HideSiblings(pNd)

- $x_ShowSiblings(pNd)
- $x_Class(pNd,pClass)
- $x_SetSiblingsClass(pNd, pClass, pNdClass)
- $x_ByClass(pClass, pNd, pTag)
- $x_ShowAllByClass(pNd, pClass, pTag)
- $x_ShowChildren(pNd)
- $x_HideChildren(pNd)
- $x_disableItem(pNd, pTest)
- $f_get_emptys(pNd, pClassFail, pClass)
- $v_Array(pNd)
- $f_ReturnChecked(pNd)
- $d_ClearAndHide(pNd)
- $f_SelectedOptions(pNd)
- $f_SelectValue(pNd)
- $u_ArrayToString(pArray, pDelim)
- $x_CheckImageSrc(pId,pSearch)
- $v_CheckValueAgainst(pThis, pValue)
- $f_Hide_On_Value_Item(pThis, pThat, pValue)
- $f_Show_On_Value_Item(pThis, pThat, pValue)
- $f_Hide_On_Value_Item_Row(pThis, pThat, pValue)
- $f_Show_On_Value_Item_Row(pThis, pThat, pValue)
- $f_DisableOnValue(pThis, pValue, pThat)
- $x_ClassByClass(pNd, pClass, pTag, pClass2)
- $f_ValuesToArray(pThis, pClass, pTag)
- $x_FormItems(pNd, pType)
- $f_CheckAll(pThis, pCheck, pArray)
- $f_CheckFirstColumn(pNd)
- $v_PopupReturn(pValue, pThat)
- $x_ToggleWithImage(pThis,pNd)
- $x_SwitchImageSrc(pNd, pSearch, pReplace)
- $x_CheckImageSrc(pNd, pSearch)
- $u_SubString(pText,pMatch)
- html_RemoveAllChildren(pNd)
- $v_IsEmpty(pThis)
- html_SetSelectValue(pId,pValue)
- addLoadEvent(pFunction)
- $f_Swap(pThis,pThat)

- submitEnter(pNd,e)
- $f_SetValueSequence(pArray,pMultiple)
- $dom_AddTag(pThis, pTag, pText)
- $tr_AddTD(pThis,pText)
- $dom_AddInput(pThis,pType,pId,pName,pValue)
- $dom_MakeParent(p_Node,p_Parent)
- $x_RowHighlight(pThis, pColor)
- $x_RowHighlightOff(pThis)
- $v_Upper(pNd)
- $v_Upper(pNd)
- $d_Find(pThis,pString,pTags,pClass)
- returnInput(p_R, p_D)
- setReturn(p_R,p_D)
- $f_First_field(pNd)
- GetCookie (pName)
- SetCookie (pName,pValue)

apex.event.trigger(pSelector,pEvent,pData)

Given a jQuery selector, jQuery object or DOM Node the specified pEvent will be triggered. pEvent can be a browser event like "click" or "change" but also a custom event like "slidechange". Returns true if the event has successfully been executed, false if not.

Return Value

```
Boolean
```

Parameters

```
pSelector (jQuery selector | jQuery object | DOM Node)
pEvent (String)
pData (Object)
```

apex.widget.initPageItem(pName,pOptions)

Given a page item, name different options that can be registered for a page item with the Application Express JavaScript framework. This is necessary to seamlessly integrate a plug-in item type with the built-in page items that are associated with Application Express JavaScript functions.

Parameters

```
pName (string)
pOptions (Object)
```

$x(pNd)

Given a DOM node or string ID (pNd), this function returns a DOM node if the element is on the page, or returns false if it is not.

Return Value

```
(DOM Node | false)
```

Parameters

```
pNd (DOM Node | string ID)
```

$v(pNd)

Given a DOM node or string ID (pNd), this function returns the value of an Application Express item in the same format as it would be posted.

Parameters

```
pNd (DOM Node | string ID)
```

$v2(pNd)

Given a DOM node or string ID (pNd), this function returns the value of an Application Express item as a string or an array. If the page item type can contain multiple values like a shuttle, checkboxes or a multi select list an array will be returned, otherwise a string.

Return Value

```
(string|array)
```

Parameters

```
pNd (DOM Node | string ID)
```

$s(pNd, pValue)

Given a DOM node or string ID (pNd), this function sets the Application Express item value taking into account what type of item it is. The pDisplayValue is optional. If used for a page item of type "Popup LOV" where the attribute "Input Field" = "Not Enterable, Show Display Value and Store Return Value". It will be used to set the "Input Field". The value of pValue will be stored in the hidden return field.

Parameters

```
pNd (DOM Node | string ID)
pValue  (String | Array)
pDisplayValue(String)
```

$u_Carray(pNd)

Given a DOM node or string ID or an array (pNd), this function returns an array. Used for creating DOM based functionality that can accept a single or multiple DOM nodes.

Return Value

```
pNd (DOM Node | string ID | Array)
```

Parameters

```
Array
```

$u_Narray(pNd)

Given a DOM node or string ID or an array (pNd), this function returns a single value, if an pNd is an array but only has one element the value of that element will be returned otherwise the array will be returned. Used for creating DOM based functionality that can accept a single or multiple DOM nodes.

Return Value

```
Array (DOM Node | string ID | Array)
```

Parameters

```
Array or first value
```

$nvl(pTest, pDefault)

If `pTest` is empty or false return `pDefault` otherwise return `pTest`.

Return Value

`(string | Array)`

Parameters

`pTest  (String | Array)`
`pDefault (String | Array)`

apex.submit(pRequest | pOptions)

This function submits the page setting the Application Express Request value (pRequest) or (pOptions).

Parameters

pRequest (String)
or
pOptions (Object)
where Object can contain the following options:

request - The request value to set (defaults to null)
set - Object conatining name/value pairs of items to be set on the page prior to submission(defaults to null).

Example

```
apex.submit({
  request:"DELETE",
  set:{"P1_DEPTNO":10, "P1_EMPNO":5433});
```

apex.confirm(pMessage, pRequest)

Displays a confirmation showing a message (`pMessage`) and depending on user's choice, submits a page setting request value (`pRequest`) or cancels page submit.

Parameters

```
pMessage (string)
pRequest (string)
```

apex.confirm(pMessage, pOptions)

Displays a confirmation showing a message (pMessage) and depending on user's choice, submits a page setting request values specified by (pOptions) or cancels page submit.

Parameters

pMessage (string)
pOptions (object)
Object can contain the following options:
request - The request value to be set (Defaults to null)
set - Object containing name / value pairs of items to be set on the page prior to submission (Defaults to null)

Example

This example shows a confirmation message with the 'Delete Department?' text. If the user chooses to proceed with the delete, the page is submitted with a REQUEST value of 'DELETE' and 2 page item values are set, P1_DEPTNO to 10 and P1_EMPNO to 5433."

```
apex.confirm("Delete Department?", {
  request:"DELETE",
  set:{"P1_DEPTNO":10, "P1_EMPNO":5433});
```

$x_Style(pNd, pStyle, pString)

Sets a specific style property (`pStyle`) to given value (`pString`) of a DOM node or DOM node Array (`pNd`).

Return Value

```
(DOM node | DOM Array)
```

Parameters

```
pNd (DOM node | string ID | DOM node Array )
pStyle (String)
pString (String)
```

$x_Hide(pNd)

Hides a DOM node or array of DOM nodes (`pNd`). This will also take into consideration which type of Application Express item is being hidden.

Return Value

`(DOM node | Array)`

Parameters

`pNd (DOM node | string ID | DOM node Array )`

$x_Show(pNd)

Shows a DOM node or array of DOM nodes (pNd). This will also take into consideration which type of Application Express item is being hidden.

Return Value

(DOM node | Array)

Parameters

pNd (DOM node | string ID | DOM node Array)

$x_Toggle(pNd)

Toggles a DOM node or array of DOM nodes (pNd).

Return Value

```
(DOM node | Array)
```

Parameters

```
pNd (DOM node | string ID | Array)
```

$x_Remove(pNd)

Removes a DOM node or array of DOM nodes.

Return Value

```
(DOM Node | Array)
```

Parameters

```
pNd (DOM node | string ID | DOM node Array)
```

$x_Value(pNd,pValue)

Sets the value (`pValue`) of a DOM node or array of DOM nodes (`pNd`).

Return Value

`Not applicable.`

Parameters

`pNd (DOM node | string ID | DOM node Array)`
`pValue (String)`

$x_UpTill(pNd, pToTag)

Starting from a DOM node (pNd), this function cascades up the DOM tree until the tag of node name (pToTag) is found.

Return Value

```
(DOM Node | false)
```

Parameters

```
pNd  (DOM Node | string ID)
String (pToTag)
String (pToClass )
```

$x_ItemRow(pNd,pFunc)

Given DOM node or array of DOM nodes, this function (shows, hides, or toggles) the entire row that contains the DOM node or array of DOM nodes. This is most useful when using Page Items.

Return Value

Not applicable.

Parameters

pNd (DOM Node | string ID | Dom node Array)
pFunc ['TOGGLE','SHOW','HIDE'] (String)

$x_HideItemRow(pNd)

Given a page item name, this function hides the entire row that holds the item. In most cases, this will be the item and its label.

Return Value

Not applicable.

Parameters

pNd (DOM Node | string ID | DON node Array)

$x_ShowItemRow(pNd)

Given a page item name, this function shows the entire row that holds the item. In most cases, this will be the item and its label.

Return Value

Not applicable.

Parameters

pNd (DOM node | string ID | DOM note Array)

$x_ToggleItemRow(pNd)

Given a page item name (pNd), this function toggles the entire row that holds the item. In most cases, this will be the item and its label.

Return Value

Not applicable.

Parameters

pNd (DOM node | string ID | DOM node ray)

$x_HideAllExcept(pNd,pNdArray)

Hides all DOM nodes referenced in `pNdArray` and then shows the DOM node referenced by `pNd`. This is most useful when `pNd` is also a node in `pNdArray`.

Return Value

```
(DOM node | DOM Array)
```

Parameters

```
pNd (DOM node | string ID | DOM node Array)
pNdArray (DOM node | String | Array)
```

$x_HideSiblings(pNd)

Hides all sibling nodes of given `pNd`.

Return Value

`(DOM node)`

Parameters

`pNd (DOM node | string ID )`

$x_ShowSiblings(pNd)

Shows all sibling DOM nodes of given DOM nodes (`pNd`).

Return Value

```
(DOM node)
```

Parameters

```
pNd (DOM node | string ID )
```

$x_Class(pNd,pClass)

Sets a DOM node or array of DOM nodes to a single class name.

Return Value

Not applicable.

Parameters

pNd (DOM node | string ID | DOM node Array)
pClass (String)

$x_SetSiblingsClass(pNd, pClass, pNdClass)

Sets the class (`pClass`) of all DOM node siblings of a node (`pNd`). If `pNdClass` is not null the class of `pNd` is set to `pNdClass`.

Return Value

```
(DOM node | false)
```

Parameters

```
pNd (DOM Nnde | string ID)
pClass (String)
pThisClass (String)
```

$x_ByClass(pClass, pNd, pTag)

Returns an array of DOM nodes by a given class name (`pClass`). If the `pNd` parameter is provided, then the returned elements will be all be children of that DOM node. Including the `pTag` parameter further narrows the list to just return nodes of that tag type.

Return Value

```
(Array)
```

Parameters

```
pClass (String)
pNd  (DOM node | string ID)
pTag (String)
```

$x_ShowAllByClass(pNd, pClass, pTag)

Show all the DOM node children of a DOM node (`pNd`) that have a specific class (`pClass`) and tag (`pTag`).

Return Value

`Not applicable.`

Parameters

```
pNd (DOM node | string ID)
pClass (String)
pTag (String)
```

$x_ShowChildren(pNd)

Show all DOM node children of a DOM node (pNd).

Return Value

Not applicable.

Parameters

pNd (DOM node | string ID)

$x_HideChildren(pNd)

Hide all DOM node children of a DOM node (`pNd`).

Return Value

`Not applicable.`

Parameters

`pNd (DOM node | string ID)`

$x_disableItem(pNd, pTest)

Disables or enables an item or array of items based on (`pTest`).

Return Value

`Not applicable.`

Parameters

`pNd (DOM node | string ID | DOM node array)`
`a (true | false)`

$f_get_emptys(pNd, pClassFail, pClass)

Checks an item or an array of items to see if any are empty, set the class of all items that are empty to `pClassFail`, set the class of all items that are not empty to `pClass`.

Return Value

```
false, Array  Array of all items that are empty (false | Array)
```

Parameters

```
pNd (DOM node | string ID | DOM node Array)
Sting (pClassFail)
Sting (pClass)
```

$v_Array(pNd)

Returns an item value as an array. Useful for multiselects and checkboxes.

Return Value

(Array)

Parameters

pId (DOM Node | string ID)

$f_ReturnChecked(pNd)

Returns an item value as an array. Useful for radio items and check boxes.

Return Value

```
(Array)
```

Parameters

```
pId (DOM node | string ID)
```

$d_ClearAndHide(pNd)

Clears the content of an DOM node or array of DOM nodes and hides them.

Return Value

Not applicable.

Parameters

pNd (DOM node | string ID | DOM node array)

$f_SelectedOptions(pNd)

Returns the DOM nodes of the selected options of a select item (pNd).

Return Value

(DOM Array)

Parameters

pNd (DOM node | string ID)

$f_SelectValue(pNd)

Returns the values of the selected options of a select item (`pNd`).

Return Value

```
(DOM Array | String)
```

Parameters

```
pNd (DOM node | string ID)
```

$u_ArrayToString(pArray, pDelim)

Given an array (`pArray`) return a string with the values of the array delimited with a given delimiter character (`pDelim`).

Return Value

`Not applicable.`

Parameters

`pArray (pArray)`
`pDelim (String)`

$x_CheckImageSrc(pId,pSearch)

Checks an image (`pId`) `source` attribute for a substring (`pSearch`). The function returns true if a substring (`pSearch`) is found. It returns false if a substring (`pSearch`) is not found.

Return Value

```
(true | false)
```

Parameters

```
pId (DOM Node | String)
pSearch (pSearch)
```

$v_CheckValueAgainst(pThis, pValue)

Checks an page item's (`pThis`) value against a set of values (`pValue`). This function returns true if any value matches.

Return Value

```
(true | false)
```

Parameters

```
pThis (DOM node | string ID)
pValue (Number | String | Array)
```

$f_Hide_On_Value_Item(pThis, pThat, pValue)

Checks an page item's (`pThis`) value against a value (`pValue`). If it matches, a DOM node (`pThat`) is set to hidden. If it does not match, then the DOM node (`pThat`) is set to visible.

Return Value

```
(true | false)
```

Parameters

```
pThis (DOM node | string ID)
pThat  (DOM node | string ID | DOM node Array )
pValue (Number | String | Array)
```

$f_Show_On_Value_Item(pThis, pThat, pValue)

Checks an page item's (`pThis`) value against a value (`pValue`). If it matches, a DOM node (`pThat`) is set to hidden. If it does not match, then the DOM node (`pThat`) is set to visible.

Return Value

```
(true | false)
```

Parameters

```
pThis (DOM node | string ID)
pThat  (DOM node | string ID | DOM node Array )
pValue (Number | String | Array)
```

$f_Hide_On_Value_Item_Row(pThis, pThat, pValue)

Checks the value (`pValue`) of an item (`pThis`). If it matches, this function hides the table row that holds (`pThat`). If it does not match, then the table row is shown.

Return Value

```
(true | false)
```

Parameters

```
pThis (DOM node | string ID)
pThat  (DOM node | string ID | DOM node Array )
pValue (Number | String | Array)
```

$f_Show_On_Value_Item_Row(pThis, pThat, pValue)

Checks the value (`pValue`) of an item (`pThis`). If it matches, this function hides the table row that holds (`pThat`). If it does not match, then the table row is shown.

Return Value

```
(true | false)
```

Parameters

```
pThis (DOM node | string ID)
pThat  (DOM node | string ID | DOM node Array )
pValue (Number | String | Array)
```

$f_DisableOnValue(pThis, pValue, pThat)

Checks the value (`pValue`) of an item (`pThis`). If it matches, this function disables the item or array of items (`pThat`). If it does not match, then the item is enabled.

Return Value

```
(true | false)
```

Parameters

```
pThis (DOM node | string ID)
pValue (String)
pThat  (DOM node | string ID | DOM node Array )
```

$x_ClassByClass(pNd, pClass, pTag, pClass2)

Sets a class attribute of an array of nodes that are selected by class.

Return Value

```
(DOM node | DOM node Array)
```

Parameters

```
pNd (DOM node | string ID)
pClass (String)
pTag (String)
pClass2 (String)
```

$f_ValuesToArray(pThis, pClass, pTag)

Collects the values of form items contained within DOM node (`pThis`) of class attribute (`pClass`) and nodeName (`pTag`) and returns an array.

Return Value

No applicable.

Parameters

pThis (DOM node | string ID)
pCLass (String)
pTag (String)

$x_FormItems(pNd, pType)

Returns all form input items contained in a DOM node (`pThis`) of a certain type (`pType`).

Return Value

```
DOM node Array
```

Parameters

```
pNd (DOM node | string ID)
pType (String)
```

$f_CheckAll(pThis, pCheck, pArray)

Check or uncheck (`pCheck`) all check boxes contained within a DOM node (`pThis`). If an array of checkboxes DOM nodes (`pArray`) is provided, use that array for affected check boxes.

Return Value

`Not applicable.`

Parameters

```
pThis (DOM node | string ID)
pCheck (true | fales)
pArray (DOM node array)
```

$f_CheckFirstColumn(pNd)

This function sets all checkboxes located in the first column of a table based on the checked state of the calling checkbox (`pNd`), useful for tabular forms.

Return Value

`DOM node Array`

Parameters

`pNd (DOM node | String)`

$v_PopupReturn(pValue, pThat)

Sets the value of the item in the parent window (`pThat`), with (`pValue`) and then closes the popup window.

Return Value

Not applicable.

Parameters

pValue (string)
pThat (DOM node | string ID)

$x_ToggleWithImage(pThis,pNd)

Given an image element (`pThis`) and a DOM node (`pNd`), this function toggles the display of the DOM node (`pNd`). The src attribute of the image element (`pThis`) will be rewritten. The image src will have any plus substrings replaced with minus substrings or minus substrings will be replaced with plus substrings.

Return Value

```
(DOM Node)
```

Parameters

```
pThis (DOM Node | string ID)
pNd (DOM Nnde | string iD | DOM node Array)
```

$x_SwitchImageSrc(pNd, pSearch, pReplace)

Checks an image (`pId`) src attribute for a substring (`pSearch`). If a substring is found, this function replaces the image entire src attribute with (`pReplace`).

Return Value

```
(DOM node | false)
```

Parameters

```
pNd (DOM node | string ID)
pSearch (String)
pReplace (String)
```

$x_CheckImageSrc(pNd, pSearch)

Checks an image (`pNd`) source attribute for a substring (`pSearch`). The function returns true if a substring (`pSearch`) is found. It returns false if a substring (`pSearch`) is not found.

Return Value

```
(true | fales)
```

Parameters

```
pNd  (DOM node | string ID)
pSearch (String)
```

$u_SubString(pText,pMatch)

Returns a true or false if a string (`pText`) contains a substring (`pMatch`).

Return Value

`(true | false)`

Parameters

`pText (String)`
`pMatch (String)`

html_RemoveAllChildren(pNd)

Use DOM methods to remove all DOM children of DOM node (pND).

Return Value

Not applicable.

Parameters

pNd (DOM node | string ID)

$v_IsEmpty(pThis)

Returns true or false if a form element is empty, this will consider any whitespace including a space, a tab, a form-feed, as empty. This will also consider any null value that has been specified on the item.

Return Value

```
[true | false]
```

Parameters

```
pThis (DOM Node | String)
```

html_SetSelectValue(pId,pValue)

Sets the value (`pValue`) of a select item (`pId`). If the value is not found, this functions selects the first option (usually the `NULL` selection).

Return Value

`Not applicable.`

Parameters

`pId (DOM node | String)`
`pValue (String)`

addLoadEvent(pFunction)

Adds an onload function (`func`) without overwriting any previously specified onload functions.

Return Value

Not applicable.

Parameters

pFunction (Javascript Function)

$f_Swap(pThis,pThat)

Swaps the form values of two form elements (`pThis`,`pThat`).

Return Value

`Not applicable.`

Parameters

`pThis (DOM Node | String)`
`pThat (DOM Node | String)`

submitEnter(pNd,e)

Submits a page when ENTER is pressed in a text field, setting the request value to the ID of a DOM node (`pNd`).

Usage is `onkeypress="submitEnter(this,event)"`

Return Value

`Not applicable.`

Parameters

`pNd (DOM node | String | Array)`

$f_SetValueSequence(pArray,pMultiple)

Sets array of form item (`pArray`) to sequential number in multiples of (`pMultiple`).

Return Value

`Not applicable.`

Parameters

`pArray (Array)`
`pMultiple (Number)`

$dom_AddTag(pThis, pTag, pText)

Inserts the html element (`pTag`) as a child node of a DOM node (`pThis`) with the innerHTML set to (`pText`).

Return Value

`DOM node`

Parameters

`pThis (DOM node | string ID )`
`pTag (String)`
`pText (String)`

$tr_AddTD(pThis,pText)

Appends a table cell to a table row (`pThis`). And sets the content to (`pText`).

Return Value

```
(DOM node)
```

Parameters

```
pThis (DOM node | string ID)
pText (String)
```

$tr_AddTH(pThis,pText)

Appends a table cell to a table row (`pThis`). And sets the content to (`pText`).

Return Value

`DOM node`

Parameters

```
pThis (DOM node | string ID)
pTest (String)
```

$dom_AddInput(pThis,pType,pId,pName,pValue)

Inserts the html form input element (`pType`) as a child node of a DOM node (`pThis`) with an id (`pId`) and name (`pName`) value set to `pValue`.

Return Value

```
(DOM node)
```

Parameters

```
pThis (DOM node | string ID)
pType (String)
pId (String)
pName (String)
pValue (String)
```

$dom_MakeParent(p_Node,p_Parent)

Takes a DOM node (p_Node) and makes it a child of DOM node (p_Parent) and then returns the DOM node (pNode).

Return Value

(DOM node)

Parameters

p_This (DOM node | string ID)
p_Parent (DOM node | string ID)

$x_RowHighlight(pThis, pColor)

Give an table row DOM element (`pThis`), this function sets the background of all table cells to a color (`pColor`). A global variable `gCurrentRow` is set to `pThis`.

Return Value

`Not applicable.`

Parameters

`pThis (DOM node | String)`
`pColor(String)`

$x_RowHighlightOff(pThis)

Give an table row Dom node (`pThis`), this function sets the background of all table cells to `NULL`.

Return Value

`Not applicable.`

Parameters

`pThis (DOM Element | String)`

$v_Upper(pNd)

Sets the value of a form item (pNd) to uppercase.

Return Value

Not applicable.

Parameters

pNd (DOM Node | String)

$d_Find(pThis,pString,pTags,pClass)

Hides child nodes of a Dom node (`pThis`) where the child node's inner HTML matches any instance of `pString`. To narrow the child nodes searched by specifying a tag name (`pTag`) or a class name (`pClass`). Note that the child node will be set to a block level element when set to visible.

Return Value

```
Not applicable.
```

Parameters

```
pThis (DOM node | String)
pString (String)
pTags (String
pClass (String)
```

returnInput(p_R, p_D)

Sets DOM node in the global variables `returnInput (p_R)` and `returnDisplay (p_D)` for use in populating items from popups.

Return Value

`Not applicable.`

Parameters

`p_R (DOM node | String)`
`p_R (DOM node | String)`

setReturn(p_R,p_D)

Sets DOM items in the global variables `returnInput` (`p_R`) and `returnDisplay` (`p_D`) for use in populating items from popups.

Return Value

`Not applicable.`

Parameters

`p_R`
`p_D`

$f_First_field(pNd)

Places the user focus on the a form item (pNd). If pNd is not found then this function places focus on the first found user editable field.

Return Value

```
true (if successful)
```

Parameters

```
pNd
```

GetCookie (pName)

Returns the value of cookie name (pName).

Return Value

Not applicable.

Parameters

pName (String)

SetCookie (pName,pValue)

Sets a cookie (`pName`) to a specified value (`pValue`).

Return Value

`Not applicable.`

Parameters

`pName (String)`
`pValue (String)`

10

APEX_PLSQL_JOB

You can use `APEX_PLSQL_JOB` package to run PL/SQL code in the background of your application. This is an effective approach for managing long running operations that do not need to complete for a user to continue working with your application.

Topics:

- About the APEX_PLSQL_JOB Package
- JOBS_ARE_ENABLED Function
- PURGE_PROCESS Procedure
- SUBMIT_PROCESS Function
- TIME_ELAPSED Function
- UPDATE_JOB_STATUS Procedure

About the APEX_PLSQL_JOB Package

APEX_PLSQL_JOB is a wrapper package around DBMS_JOB functionality offered in the Oracle database. Note that the APEX_PLSQL_JOB package only exposes that functionality which is necessary to run PL/SQL in the background.

Table 10–1 describes the functions available in the APEX_PLSQL_JOB package.

Table 10–1 APEX_PLSQL_JOB Package: Available Functions

Function or Procedure	Description
SUBMIT_PROCESS	Use this procedure to submit background PL/SQL. This procedure returns a unique job number. Because you can use this job number as a reference point for other procedures and functions in this package, it may be useful to store it in your own schema.
UPDATE_JOB_STATUS	Call this procedure to update the status of the currently running job. This procedure is most effective when called from the submitted PL/SQL.
TIME_ELAPSED	Use this function to determine how much time has elapsed since the job was submitted.
JOBS_ARE_ENABLED	Call this function to determine whether the database is currently in a mode that supports submitting jobs to the APEX_PLSQL_JOB package.
PURGE_PROCESS	Call this procedure to clean up submitted jobs. Submitted jobs stay in the APEX_PLSQL_JOBS view until either Oracle Application Express cleans out those records, or you call PURGE_PROCESS to manually remove them.

You can view all jobs submitted to the APEX_PLSQL_JOB package using the APEX_PLSQL_JOBS view.

JOBS_ARE_ENABLED Function

Call this function to determine whether or not the database is currently in a mode that supports submitting jobs to the APEX_PLSQL_JOB package.

Syntax

```
APEX_PLSQL_JOB.JOBS_ARE_ENABLED
RETURN BOOLEAN;
```

Parameters

None.

Example

The following example shows how to use the `JOBS_ARE_ENABLED` function. In the example, if the function returns `TRUE` the message 'Jobs are enabled on this database instance' is displayed, otherwise the message 'Jobs are not enabled on this database instance' is displayed.

```
BEGIN
    IF APEX_PLSQL_JOB.JOBS_ARE_ENABLED THEN
        HTP.P('Jobs are enabled on this database instance.');
    ELSE
        HTP.P('Jobs are not enabled on this database instance.');
    END IF;
END;
```

PURGE_PROCESS Procedure

Call this procedure to clean up submitted jobs. Submitted jobs stay in the APEX_PLSQL_JOBS view until either Oracle Application Express cleans out those records, or you call PURGE_PROCESS to manually remove them.

Syntax

```
APEX_PLSQL_JOB.PURGE_PROCESS (
    p_job IN NUMBER);
```

Parameters

Table 10–2 describes the parameters available in the PURGE_PROCESS procedure.

Table 10–2 PURGE_PROCESS Parameters

Parameter	Description
p_job	The job number that identifies the submitted job you wish to purge.

Example

The following example shows how to use the PURGE_PROCESS procedure to purge the submitted job identified by a job number of 161. You could also choose to purge all or some of the current submitted jobs by referencing the APEX_PLSQL_JOBS view.

```
BEGIN
    APEX_PLSQL_JOB.PURGE_PROCESS(
        p_job => 161);
END;
```

SUBMIT_PROCESS Function

Use this function to submit background PL/SQL. This function returns a unique job number. Because you can use this job number as a reference point for other procedures and functions in this package, it may be useful to store it in your own schema.

Syntax

```
APEX_PLSQL_JOB.SUBMIT_PROCESS (
    p_sql IN VARCHAR2,
    p_when IN DATE DEFAULT SYSDATE,
    p_status IN VARCHAR2 DEFAULT 'PENDING')
RETURN NUMBER;
```

Parameters

Table 10–3 describes the parameters available in the SUBMIT_PROCESS function.

Table 10–3 SUBMIT_PROCESS Parameters

Parameter	Description
p_sql	The process you wish to run in your job. This can be any valid anonymous block, for example: 'BEGIN <your code> END;' or 'DECLARE <your declaration> BEGIN <your code> END;'
p_when	When you want to run it. The default is SYSDATE which means the job will run as soon as possible. You can also set the job to run in the future, for example: sysdate + 1 - The job will run in 1 days time. sysdate + (1/24) - The job will run in 1 hours time. sysdate + (10/24/60) - The job will run in 10 minutes time.
p_status	Plain text status information for this job.

Example

The following example shows how to use the SUBMIT_PROCESS function to submit a background process that will start as soon as possible.

```
DECLARE
    l_sql VARCHAR2(4000);
    l_job NUMBER;
BEGIN
    l_sql := 'BEGIN MY_PACKAGE.MY_PROCESS; END;';
    l_job := APEX_PLSQL_JOB.SUBMIT_PROCESS(
        p_sql => l_sql,
        p_status => 'Background process submitted');
    --store l_job for later reference
END;
```

TIME_ELAPSED Function

Use this function to determine how much time has elapsed since the job was submitted.

Syntax

```
APEX_PLSQL_JOB.TIME_ELAPSED(
    p_job IN NUMBER)
RETURN NUMBER;
```

Parameters

Table 10–4 describes the parameters available in the `TIME_ELAPSED` function.

Table 10–4 TIME_ELAPSED Parameters

Parameter	Description
`p_job`	The job ID for the job you wish to see how long since it was submitted.

Example

The following example shows how to use the `TIME_ELAPSED` function to get the time elapsed for the submitted job identified by the job number 161.

```
DECLARE
    l_time NUMBER;
BEGIN
    l_time := APEX_PLSQL_JOB.TIME_ELAPSED(p_job => 161);
END;
```

UPDATE_JOB_STATUS Procedure

Call this procedure to update the status of the currently running job. This procedure is most effective when called from the submitted PL/SQL.

Syntax

```
APEX_PLSQL_JOB.UPDATE_JOB_STATUS (
    p_job IN NUMBER,
    p_status IN VARCHAR2);
```

Parameters

Table 10–5 describes the parameters available in the UPDATE_JOB_STATUS procedure.

Table 10–5 UPDATE_JOB_STATUS Parameters

Parameter	Description
p_job	The job ID for the job you want to update the status of.
p_status	The string of up to 100 characters to be used as the current status of the job.

Example

The following example shows how to use the UPDATE_JOB_STATUS procedure. In this example, note that:

- Lines 002 to 010 run a loop that inserts 100 records into the emp table.
- APP_JOB is referenced as a bind variable inside the VALUES clause of the INSERT, and specified as the p_job parameter value in the call to UPDATE_JOB_STATUS.
- APP_JOB represents the job number which will be assigned to this process as it is submitted to APEX_PLSQL_JOB. By specifying this reserved item inside your process code, it will be replaced for you at execution time with the actual job number.
- Note that this example calls to UPDATE_JOB_STATUS every ten records, inside the block of code. Normally, Oracle transaction rules dictate updates made inside code blocks will not be seen until the entire transaction is committed. The APEX_PLSQL_JOB.UPDATE_JOB_STATUS procedure, however, has been implemented in such a way that the update will happen regardless of whether or not the job succeeds or fails. This last point is important for two reasons:

  1. Even if your status shows "100 rows inserted", it does not mean the entire operation was successful. If an error occurred at the time the block of code tried to commit, the user_status column of APEX_PLSQL_JOBS would not be affected because status updates are committed separately.
  2. Updates are performed autonomously. You can view the job status before the job has completed. This gives you the ability to display status text about ongoing operations in the background as they are happening.

```
BEGIN
    FOR i IN 1 .. 100 LOOP
        INSERT INTO emp(a,b) VALUES (:APP_JOB,i);
        IF MOD(i,10) = 0 THEN
```

```
            APEX_PLSQL_JOB.UPDATE_JOB_STATUS(
                P_JOB => :APP_JOB,
                P_STATUS => i || ' rows inserted');
        END IF;
        APEX_UTIL.PAUSE(2);
    END LOOP;
END;
```

11

APEX_LANG

You can use `APEX_LANG` API to translate messages.

Topics:

- LANG Function
- MESSAGE Function

LANG Function

This function is used to return a translated text string for translations defined in dynamic translations.

Syntax

```
APEX_LANG.LANG (
    p_primary_text_string IN VARCHAR2 DEFAULT NULL,
    p0 IN VARCHAR2 DEFAULT NULL,
    p1 IN VARCHAR2 DEFAULT NULL,
    p2 IN VARCHAR2 DEFAULT NULL,
    ...
    p9 IN VARCHAR2 DEFAULT NULL,
    p_primary_language IN VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;
```

Parameters

Table 11–1 describes the parameters available in the `APEX_LANG.LANG` function.

Table 11–1 LANG Parameters

Parameter	Description
`p_primary_text_string`	Text string of the primary language. This will be the value of the Translate From Text in the dynamic translation.
`p0` through `p9`	Dynamic substitution value: p0 corresponds to 0% in the translation string; p1 corresponds to 1% in the translation string; p2 corresponds to 2% in the translation string, and so on.
`p_primary_language`	Language code for the message to be retrieved. If not specified, Oracle Application Express uses the current language for the user as defined in the Application Language Derived From attribute. See also: Specifying the Primary Language for an Application in the *Oracle Application Express Application Builder User's Guide*.

Example

Suppose you have a table that defines all primary colors. You could define a dynamic message for each color and then apply the LANG function to the defined values in a query. For example:

```
SELECT APEX_LANG.LANG(color)
FROM my_colors
```

If you were running the application in German, RED was a value for the color column in the `my_colors` table, and you defined the German word for red, the previous example would return ROT.

MESSAGE Function

Use this function to translate text strings (or messages) generated from PL/SQL stored procedures, functions, triggers, packaged procedures, and functions.

Syntax

```
APEX_LANG.MESSAGE (
    p_name IN VARCHAR2 DEFAULT NULL,
    p0 IN VARCHAR2 DEFAULT NULL,
    p1 IN VARCHAR2 DEFAULT NULL,
    p2 IN VARCHAR2 DEFAULT NULL,
    ...
    p9 IN VARCHAR2 DEFAULT NULL,
    p_lang IN VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;
```

Parameters

Table 11–2 describes the parameters available in the `APEX_LANG.MESSAGE` function.

Table 11–2 MESSAGE Parameters

Parameter	Description
`p_name`	Name of the message as defined in Shared Components > Text Messages of your application in Oracle Application Express.
`p0` through `p9`	Dynamic substitution value: p0 corresponds to 0% in the translation string; p1 corresponds to 1% in the translation string; p2 corresponds to 2% in the translation string, and so on.
`p_lang`	Language code for the message to be retrieved. If not specified, Oracle Application Express uses the current language for the user as defined in the Application Language Derived From attribute. See also: Specifying the Primary Language for an Application in the *Oracle Application Express Application Builder User's Guide*.

Example

The following example assumes you have defined a message called GREETING_MSG in your application in English as Good morning%0 and in German as Guten Tag%1. The following example demonstrates how you could invoke this message from PL/SQL:

```
BEGIN
--
-- Print the greeting
--
APEX_LANG.MESSAGE('GREETING_MSG', V('APP_USER'));
END;
```

How the `p_lang` attribute is defined depends on how the Application Express engine derives the Application Primary Language. For example, if you are running the application in German and the previous call is made to the `APEX_LANG.MESSAGE` API, the Application Express engine first looks for a message called `GREETING_MSG` with a `LANG_CODE` of `de`. If it does not find anything, then it will revert to the Application Primary Language attribute. If it still does not find anything, the

Application Express engine looks for a message by this name with a language code of en.

> **See also:** Specifying the Primary Language for an Application in the *Oracle Application Express Application Builder User's Guide*.

12

APEX_COLLECTION

Collections enable you to temporarily capture one or more nonscalar values. You can use collections to store rows and columns currently in session state so they can be accessed, manipulated, or processed during a user's specific session. You can think of a collection as a bucket in which you temporarily store and name rows of information.

Topics:

- About the APEX_COLLECTION API
- ADD_MEMBER Procedure
- ADD_MEMBER Function
- ADD_MEMBERS Procedure
- COLLECTION_EXISTS Function
- COLLECTION_HAS_CHANGED Function
- COLLECTION_MEMBER_COUNT Function
- CREATE_COLLECTION Procedure
- CREATE_OR_TRUNCATE_COLLECTION Procedure
- CREATE_COLLECTION_FROM_QUERY Procedure
- CREATE_COLLECTION_FROM_QUERY2 Procedure
- CREATE_COLLECTION_FROM_QUERY_B Procedure
- CREATE_COLLECTION_FROM_QUERYB2 Procedure
- DELETE_ALL_COLLECTIONS Procedure
- DELETE_ALL_COLLECTIONS_SESSION Procedure
- DELETE_COLLECTION Procedure
- DELETE_MEMBER Procedure
- DELETE_MEMBERS Procedure
- GET_MEMBER_MD5 Function
- MERGE_MEMBERS Procedure
- MOVE_MEMBER_DOWN Procedure
- MOVE_MEMBER_UP Procedure
- RESEQUENCE_COLLECTION Procedure
- RESET_COLLECTION_CHANGED Procedure

- RESET_COLLECTION_CHANGED_ALL Procedure
- SORT_MEMBERS Procedure
- TRUNCATE_COLLECTION Procedure
- UPDATE_MEMBER Procedure
- UPDATE_MEMBERS Procedure
- UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1
- UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2
- UPDATE_MEMBER_ATTRIBUTE Procedure Signature 3
- UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4
- UPDATE_MEMBER_ATTRIBUTE Procedure Signature 5
- UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6

About the APEX_COLLECTION API

Every collection contains a named list of data elements (or members) which can have up to 50 character attributes (`VARCHAR2(4000)`), five number attributes, five date attributes, one XML Type attribute, one large binary attribute (`BLOB`), and one large character attribute (`CLOB`). You insert, update, and delete collection information using the PL/SQL API `APEX_COLLECTION`.

The following are examples of when you might use collections:

- When you are creating a data-entry wizard in which multiple rows of information first need to be collected within a logical transaction. You can use collections to temporarily store the contents of the multiple rows of information, before performing the final step in the wizard when both the physical and logical transactions are completed.
- When your application includes an update page on which a user updates multiple detail rows on one page. The user can make many updates, apply these updates to a collection and then call a final process to apply the changes to the database.
- When you are building a wizard where you are collecting an arbitrary number of attributes. At the end of the wizard, the user then performs a task that takes the information temporarily stored in the collection and applies it to the database.

Topics:

- Naming, Creating and Accessing Collections
- Merging, Truncating and Deleting Collections
- Adding, Updating and Deleting Collection Members
- Managing Collections

Naming, Creating and Accessing Collections

Topics:

- Naming Collections
- Creating a Collection
- About the Parameter p_generate_md5

Naming Collections

When you create a collection, you must give it a name that cannot exceed 255 characters. Note that collection names are not case-sensitive and will be converted to uppercase.

Once the collection is named, you can access the values in the collection by running a SQL query against the view `APEX_COLLECTIONS`.

See Also: "Accessing a Collection" on page 12-5, " CREATE_COLLECTION Procedure Parameters" on page 12-21, " CREATE_OR_TRUNCATE_COLLECTION Procedure Parameters" on page 12-22

Creating a Collection

Every collection contains a named list of data elements (or members) which can have up to 50 character attributes (`VARCHAR2(4000)`), five number attributes, one XML Type attribute, one large binary attribute (`BLOB`), and one large character attribute (`CLOB`). You use the following methods to create a collection:

- `CREATE_COLLECTION`
  This method creates an empty collection with the provided name. An exception is raised if the named collection exists.
- `CREATE_OR_TRUNCATE_COLLECTION`
  If the provided named collection does not exist, this method creates an empty collection with the given name. If the named collection exists, this method truncates it. Truncating a collection empties it, but leaves it in place.
- `CREATE_COLLECTION_FROM_QUERY`
  This method creates a collection and then populates it with the results of a specified query. An exception is raised if the named collection exists. This method can be used with a query with up to 50 columns in the `SELECT` clause. These columns in the `SELECT` clause will populate the 50 character attributes of the collection (C001 through C050).
- `CREATE_COLLECTION_FOM_QUERY2`
  This method creates a collection and then populates it with the results of a specified query. An exception is raised if the named collection exists. It is identical to the `CREATE_COLLECTION_FROM_QUERY`, however, the first 5 columns of the `SELECT` clause must be numeric. After the numeric columns, there can be up to 50 character columns in the `SELECT` clause.
- `CREATE_COLLECTION_FROM_QUERY_B`
  This method offers significantly faster performance than the `CREATE_COLLECTION_FROM_QUERY` method by performing bulk SQL operations, but has the following limitations:

- No column value in the select list of the query can be more than 2,000 bytes. If a row is encountered that has a column value of more than 2,000 bytes, an error will be raised during execution.
- The MD5 checksum will not be computed for any members in the collection.

- CREATE_COLLECTION_FROM_QUERYB2
  This method also creates a collection and then populates it with the results of a specified query. An exception is raised if the named collection exists. It is identical to the CREATE_COLLECTION_FROM_QUERY_B, however, the first five columns of the SELECT clause must be numeric. After the numeric columns, there can be up to 50 character columns in the SELECT clause.

  See Also: "CREATE_COLLECTION Procedure" on page 12-21, "CREATE_OR_TRUNCATE_COLLECTION Procedure" on page 12-22, "CREATE_COLLECTION_FROM_QUERY Procedure" on page 12-23, "CREATE_COLLECTION_FROM_QUERY2 Procedure" on page 12-25, "CREATE_COLLECTION_FROM_QUERY_B Procedure" on page 12-27, "CREATE_COLLECTION_FROM_QUERYB2 Procedure" on page 12-29

About the Parameter p_generate_md5

Use the p_generate_md5 flag to specify if the message digest of the data of the collection member should be computed. By default, this flag is set to NO. Use this parameter to check the MD5 of the collection member (that is, compare it with another member or see if a member has changed).

See Also: "Determining Collection Status" on page 12-11 for information about using the GET_MEMBER_MD5 function, "GET_MEMBER_MD5 Function" on page 12-36

Accessing a Collection

You can access the members of a collection by querying the database view APEX_COLLECTIONS. The APEX_COLLECTIONS view has the following definition:

```
COLLECTION_NAME   NOT NULL VARCHAR2(255)
SEQ_ID            NOT NULL NUMBER
C001              VARCHAR2(4000)
C002              VARCHAR2(4000)
C003              VARCHAR2(4000)
C004              VARCHAR2(4000)
C005              VARCHAR2(4000)
...
C050              VARCHAR2(4000)
N001              NUMBER
N002              NUMBER
N003              NUMBER
N004              NUMBER
N005              NUMBER
CLOB001           CLOB
BLOB001           BLOB
XMLTYPE001        XMLTYPE
MD5_ORIGINAL      VARCHAR2(4000)
```

Use the APEX_COLLECTIONS view in an application just as you would use any other table or view in an application, for example:

```
SELECT c001, c002, c003, n001, clob001
```

```
  FROM APEX_collections
WHERE collection_name = 'DEPARTMENTS'
```

Merging, Truncating and Deleting Collections

Topics:

- Merging Collections
- Truncating a Collection
- Deleting a Collection
- Deleting All Collections for the Current Application
- Deleting All Collections in the Current Session

Merging Collections

You can merge members of a collection with values passed in a set of arrays. By using the `p_init_query` argument, you can create a collection from the supplied query.

See Also: "MERGE_MEMBERS Procedure" on page 12-37

Truncating a Collection

If you truncate a collection, you remove all members from the specified collection, but the named collection remains in place.

See Also: "TRUNCATE_COLLECTION Procedure" on page 12-45

Deleting a Collection

If you delete a collection, you delete the collection and all of its members. Be aware that if you do not delete a collection, it will eventually be deleted when the session is purged.

See Also: "DELETE_COLLECTION Procedure" on page 12-33

Deleting All Collections for the Current Application

Use the `DELETE_ALL_COLLECTIONS` method to delete all collections defined in the current application.

See Also: "DELETE_ALL_COLLECTIONS Procedure" on page 12-31

Deleting All Collections in the Current Session

Use the `DELETE_ALL_COLLECTIONS_SESSION` method to delete all collections defined in the current session.

See Also: "DELETE_ALL_COLLECTIONS_SESSION Procedure" on page 12-32

Adding, Updating and Deleting Collection Members

Topics:

- Adding Members to a Collection
- About the Parameters p_generate_md5, p_clob001, p_blob001, and p_xmltype001
- Updating Collection Members
- Deleting Collection Members

Adding Members to a Collection

When data elements (or members) are added to a collection, they are assigned a unique sequence ID. As you add members to a collection, the sequence ID will change in increments of 1, with the newest members having the largest ID.

You add new members to a collection using the `ADD_MEMBER function`. Calling this function returns the sequence ID of the newly added member.

You can also add new members (or an array of members) to a collection using the `ADD_MEMBERS` procedure. The number of members added is based on the number of elements in the first array.

> **See Also:** "ADD_MEMBER Procedure" on page 12-12, "ADD_MEMBER Function" on page 12-14, "ADD_MEMBERS Procedure" on page 12-16

About the Parameters p_generate_md5, p_clob001, p_blob001, and p_xmltype001

Use the `p_generate_md5` flag to specify if the message digest of the data of the collection member should be computed. By default, this flag is set to `NO`. Use this parameter to check the MD5 of the collection member (that is, compare it with another member or see if a member has changed).

Use `p_clob001` for collection member attributes which exceed 4,000 characters. Use `p_blob001` for binary collection member attributes. Use `p_xmltype001` to store well-formed XML.

> **See Also:** "Determining Collection Status" on page 12-11 for information about using the function `GET_MEMBER_MD5`

Updating Collection Members

You can update collection members by calling the `UPDATE_MEMBER` procedure and referencing the desired collection member by its sequence ID. The `UPDATE_MEMBER` procedure replaces an entire collection member, not individual member attributes.

Use the `p_clob001` parameter for collection member attributes which exceed 4,000 characters.

To update a single attribute of a collection member, use the `UPDATE_MEMBER_ATTRIBUTE procedure`.

See Also: "UPDATE_MEMBER Procedure" on page 12-46, "UPDATE_MEMBERS Procedure" on page 12-48, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1" on page 12-50, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2" on page 12-52, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 3" on page 12-54, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4" on page 12-56, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 5" on page 12-58

Deleting Collection Members

You can delete a collection member by calling the `DELETE_MEMBER` procedure and referencing the desired collection member by its sequence ID. Note that this procedure leaves a gap in the sequence IDs in the specified collection.

You can also delete all members from a collection by when an attribute matches a specific value. Note that the `DELETE_MEMBERS` procedure also leaves a gap in the sequence IDs in the specified collection. If the supplied attribute value is null, then all members of the named collection will be deleted where the attribute (specified by `p_attr_number`) is null.

See Also: "DELETE_MEMBER Procedure" on page 12-34, "DELETE_MEMBERS Procedure" on page 12-35

Managing Collections

Topics:

- Obtaining a Member Count
- Resequencing a Collection
- Verifying Whether a Collection Exists
- Adjusting a Member Sequence ID
- Sorting Collection Members
- Clearing Collection Session State

Obtaining a Member Count

Use `COLLECTION_MEMBER_COUNT` to return the total count of all members in a collection. Note that this count does not indicate the highest sequence in the collection.

See Also: "COLLECTION_MEMBER_COUNT Function" on page 12-20

Resequencing a Collection

Use `RESEQUENCE_COLLECTION` to resequence a collection to remove any gaps in sequence IDs while maintaining the same element order.

See Also: "RESEQUENCE_COLLECTION Procedure" on page 12-41

Verifying Whether a Collection Exists

Use `COLLECTION_EXISTS` to determine if a collection exists.

See Also: "COLLECTION_EXISTS Function" on page 12-18

Adjusting a Member Sequence ID

You can adjust the sequence ID of a specific member within a collection by moving the ID up or down. When you adjust a sequence ID, the specified ID is exchanged with another ID. For example, if you were to move the ID 2 up, 2 becomes 3, and 3 would become 2.

Use `MOVE_MEMBER_UP` to adjust a member sequence ID up by one. Alternately, use `MOVE_MEMBER_DOWN` to adjust a member sequence ID down by one.

See Also: "MOVE_MEMBER_DOWN Procedure" on page 12-39, "MOVE_MEMBER_UP Procedure" on page 12-40

Sorting Collection Members

Use the `SORT_MEMBERS` method to reorder members of a collection by the column number. This method sorts the collection by a particular column number and also reassigns the sequence IDs for each member to remove gaps.

See Also: "SORT_MEMBERS Procedure" on page 12-44

Clearing Collection Session State

Clearing the session state of a collection removes the collection members. A shopping cart is a good example of when you might need to clear collection session state. When

a user requests to empty the shopping cart and start again, you must clear the session state for a collection. You can remove session state of a collection by calling the TRUNCATE_COLLECTION method or by using f?p syntax.

Calling the TRUNCATE_COLLECTION method deletes the existing collection and then recreates it, for example:

```
APEX_COLLECTION.TRUNCATE_COLLECTION(
    p_collection_name => collection name);
```

You can also use the sixth f?p syntax argument to clear session state, for example:

```
f?p=App:Page:Session::NO:collection name
```

See Also: "TRUNCATE_COLLECTION Procedure" on page 12-45

Determining Collection Status

The p_generate_md5 parameter determines if the MD5 message digests are computed for each member of a collection. The collection status flag is set to FALSE immediately after you create a collection. If any operations are performed on the collection (such as add, update, truncate, and so on), this flag is set to TRUE.

You can reset this flag manually by calling RESET_COLLECTION_CHANGED.

Once this flag has been reset, you can determine if a collection has changed by calling COLLECTION_HAS_CHANGED.

When you add a new member to a collection, an MD5 message digest is computed against all 50 attributes and the CLOB attribute if the p_generated_md5 parameter is set to YES. You can access this value from the MD5_ORIGINAL column of the view APEX_COLLECTION. You can access the MD5 message digest for the current value of a specified collection member by using the function GET_MEMBER_MD5.

See Also: "RESET_COLLECTION_CHANGED Procedure" on page 12-42, "COLLECTION_HAS_CHANGED Function" on page 12-19, "GET_MEMBER_MD5 Function" on page 12-36

ADD_MEMBER Procedure

Use this procedure to add a new member to an existing collection. An error is raised if the specified collection does not exist for the current user in the same session for the current Application ID. Gaps are not used when adding a new member, so an existing collection with members of sequence IDs (1,2,5,8) will add the new member with a sequence ID of 9.

Syntax

```
APEX_COLLECTION.ADD_MEMBER (
    p_collection_name IN VARCHAR2,
    p_c001 IN VARCHAR2 default null,
    ...
    p_c050 IN VARCHAR2 default null,
    p_n001 IN NUMBER default null,
    p_n002 IN NUMBER default null,
    p_n003 IN NUMBER default null,
    p_n004 IN NUMBER default null,
    p_n005 IN NUMBER default null,
    p_d001 IN DATE default null,
    p_d002 IN DATE default null,
    p_d003 IN DATE default null,
    p_d004 IN DATE default null,
    p_d005 IN DATE default null,
    p_clob001 IN CLOB default empty_clob(),
    p_blob001 IN BLOB default empty_blob(),
    p_xmltype001 IN XMLTYPE default null,
    p_generate_md5 IN VARCHAR2  default 'NO');
```

Parameters

Table 12–1 describes the parameters available in the `ADD_MEMBER` procedure.

Note: Any character attribute exceeding 4,000 characters is truncated to 4,000 characters.

Table 12–1 ADD_MEMBER Procedure Parameters

Parameter	Description
`p_collection_name`	The name of an existing collection. Maximum length is 255 bytes. Collection names are not case sensitive and are converted to upper case.
`p_c001 through p_c050`	Attribute value of the member to be added. Maximum length is 4,000 bytes. Any character attribute exceeding 4,000 characters is truncated to 4,000 characters.
`p_n001 through p_n005`	Attribute value of the numeric attributes to be added.
`p_d001` through `p_d005`	Attribute value of the date attribute.
`p_clob001`	Use p_clob001 for collection member attributes that exceed 4,000 characters.
`p_blob001`	Use p_blob001 for binary collection member attributes.
`p_xmltype001`	Use p_xmltype001 to store well-formed XML.

Table 12–1 (Cont.) ADD_MEMBER Procedure Parameters

Parameter	Description
p_generate_md5	Valid values include YES and NO. YES to specify if the message digest of the data of the collection member should be computed. Use this parameter to compare the MD5 of the collection member with another member or to see if that member has changed.

Example

The following is an example of the ADD_MEMBER procedure.

```
APEX_COLLECTION.ADD_MEMBER(
        p_collection_name => 'GROCERIES'
        p_c001            => 'Grapes',
        p_c002            => 'Imported',
        p_n001            => 125,
        p_d001            => sysdate );
END;
```

See Also: "GET_MEMBER_MD5 Function" on page 12-36, "ADD_MEMBER Function" on page 12-14, "ADD_MEMBERS Procedure" on page 12-16

ADD_MEMBER Function

Use this function to add a new member to an existing collection. Calling this function returns the sequence ID of the newly added member. An error is raised if the specified collection does not exist for the current user in the same session for the current Application ID. Gaps are not used when adding a new member, so an existing collection with members of sequence IDs (1,2,5,8) will add the new member with a sequence ID of 9.

Syntax

```
APEX_COLLECTION.ADD_MEMBER (
    p_collection_name IN VARCHAR2,
    p_c001 IN VARCHAR2 default null,
    ...
    p_c050 IN VARCHAR2 default null,
    p_n001 IN NUMBER default null,
    p_n002 IN NUMBER default null,
    p_n003 IN NUMBER default null,
    p_n004 IN NUMBER default null,
    p_n005 IN NUMBER default null,
    p_d001 IN DATE default null,
    p_d002 IN DATE default null,
    p_d003 IN DATE default null,
    p_d004 IN DATE default null,
    p_d005 IN DATE default null,
    p_clob001 IN CLOB default empty_clob(),
    p_blob001 IN BLOB default empty_blob(),
    p_xmltype001 IN XMLTYPE default null,
    p_generate_md5 IN VARCHAR2 default 'NO')
RETURN NUMBER;
```

Parameters

Table 12–2 describes the parameters available in the `ADD_MEMBER` function.

> **Note:** Any character attribute exceeding 4,000 characters is truncated to 4,000 characters.

Table 12–2 ADD_MEMBER Function Parameters

Parameter	Description
`p_collection_name`	The name of an existing collection. Maximum length is 255 bytes. Collection names are not case sensitive and are converted to upper case.
`p_c001 through p_c050`	Attribute value of the member to be added. Maximum length is 4,000 bytes. Any character attribute exceeding 4,000 characters is truncated to 4,000 characters.
`p_n001 through p_n005`	Attribute value of the numeric attributes to be added.
`p_d001` through `p_d005`	Attribute value of the date attribute to be added.
`p_clob001`	Use p_clob001 for collection member attributes that exceed 4,000 characters.
`p_blob001`	Use p_blob001 for binary collection member attributes.

Table 12–2 (Cont.) ADD_MEMBER Function Parameters

Parameter	Description
p_xmltype001	Use p_xmltype001 to store well-formed XML.
p_generate_md5	Valid values include YES and NO. YES to specify if the message digest of the data of the collection member should be computed. Use this parameter to compare the MD5 of the collection member with another member or to see if that member has changed.

Example

```
DECLARE
    l_seq number;
BEGIN
    l_seq := APEX_COLLECTION.ADD_MEMBER(
                p_collection_name => 'GROCERIES'
                p_c001            => 'Grapes',
                p_c002            => 'Imported',
                p_n001            => 125,
                p_d001            => sysdate );
END;
```

See Also: "GET_MEMBER_MD5 Function" on page 12-36, "ADD_MEMBER Procedure" on page 12-12, "ADD_MEMBERS Procedure" on page 12-16

ADD_MEMBERS Procedure

Use this procedure to add an array of members to a collection. An error is raised if the specified collection does not exist for the current user in the same session for the current Application ID. Gaps are not used when adding a new member, so an existing collection with members of sequence IDs (1,2,5,8) will add the new member with a sequence ID of 9. The count of elements in the p_c001 PL/SQL table will be used as the total number of items across all PL/SQL tables. For example, if p_c001.count is 2 and p_c002.count is 10, only 2 members will be added. If p_c001 is null an application error is raised.

Syntax

```
APEX_COLLECTION.ADD_MEMBERS (
    p_collection_name IN VARCHAR2,
    p_c001 IN APEX_APPLICATION_GLOBAL.VC_ARR2 defulat empty_vc_arr,
    p_c002 IN APEX_APPLICATION_GLOBAL.VC_ARR2 defulat empty_vc_arr,
    p_c003 IN APEX_APPLICATION_GLOBAL.VC_ARR2 defulat empty_vc_arr,
    ...
    p_c050 IN APEX_APPLICATION_GLOBAL.VC_ARR2 defulat empty_vc_arr,
    p_n001 IN APEX_APPLICATION_GLOBAL.N_ARR defulat empty_n_arr,
    p_n002 IN APEX_APPLICATION_GLOBAL.N_ARR defulat empty_n_arr,
    p_n003 IN APEX_APPLICATION_GLOBAL.N_ARR defulat empty_n_arr,
    p_n004 IN APEX_APPLICATION_GLOBAL.N_ARR defulat empty_n_arr,
    p_n005 IN APEX_APPLICATION_GLOBAL.N_ARR defulat empty_n_arr,
    p_generate_md5 IN VARCHAR2 default 'NO');
```

Parameters

Table 12–3 describes the parameters available in the `ADD_MEMBERS` procedure.

> **Note:** Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. Also, the number of members added is based on the number of elements in the first array.

Table 12–3 ADD_MEMBERS Procedure Parameters

Parameter	Description
`p_collection_name`	The name of an existing collection. Maximum length is 255 bytes. Collection names are not case sensitive and are converted to upper case.
`p_c001 through p_c050`	Attribute value of the member to be added. Maximum length is 4,000 bytes. Any character attribute exceeding 4,000 characters is truncated to 4,000 characters.
`p_generate_md5`	Valid values include `YES` and `NO`. `YES` to specify if the message digest of the data of the collection member should be computed. Use this parameter to compare the MD5 of the collection member with another member or to see if that member has changed.

Example

The following example shows how to add two new members to the `EMPLOYEE` table.

```
Begin
```

```
    APEX_COLLECTION.ADD_MEMBERS(
        p_collection_name => 'EMPLOYEE',
        p_c001 => l_arr1,
        p_c002 => l_arr2);
End;
```

See Also: "GET_MEMBER_MD5 Function" on page 12-36, "ADD_MEMBER Procedure" on page 12-12, "ADD_MEMBER Function" on page 12-14

COLLECTION_EXISTS Function

Use this function to determine if a collection exists. A `TRUE` is returned if the specified collection exists for the current user in the current session for the current Application ID, otherwise `FALSE` is returned.

Syntax

```
APEX_COLLECTION.COLLECTION_EXISTS (
    p_collection_name IN VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 12–4 describes the parameters available in the `COLLECTION_EXISTS` function.

Table 12–4 COLLECTION_EXISTS Function Parameters

Parameter	Description
`p_collection_name`	The name of the collection. Maximum length is 255 bytes. The collection name is not case sensitive and is converted to upper case.

Example

The following example shows how to use the `COLLECTION_EXISTS` function to determine if the collection named `EMPLOYEES` exists.

```
Begin
    l_exists := APEX_COLLECTION.COLLECTION_EXISTS (
        p_collection_name => 'EMPLOYEES';
End;
```

COLLECTION_HAS_CHANGED Function

Use this function to determine if a collection has changed since it was created or the collection changed flag was reset.

Syntax

```
APEX_COLLECTION.COLLECTION_HAS_CHANGED (
    p_collection_name IN VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 12–5 describes the parameters available in the COLLECTION_HAS_CHANGED function.

Table 12–5 COLLECTION_HAS_CHANGED Function Parameters

Parameter	Description
p_collection_name	The name of the collection. An error is returned if this collection does not exist with the specified name of the current user and in the same session.

Example

The following example shows how to use the COLLECTION_HAS_CHANGED function to determine if the EMPLOYEES collection has changed since it was created or last reset.

```
Begin
    l_exists := APEX_COLLECTION.COLLECTION_HAS_CHANGED (
        p_collection_name => 'EMPLOYEES';
End;
```

COLLECTION_MEMBER_COUNT Function

Use this function to get the total number of members for the named collection. If gaps exist, the total member count returned is not equal to the highest sequence ID in the collection. If the named collection does not exist for the current user in the current session, an error is raised.

Syntax

```
APEX_COLLECTION.COLLECTION_MEMBER_COUNT (
    p_collection_name IN VARCHAR2)
RETURN NUMBER;
```

Parameters

Table 12–6 describes the parameters available in the COLLECTION_MEMBER_COUNT function.

Table 12–6 COLLECTION_MEMBER_COUNT Function Parameters

Parameter	Description
p_collection_name	The name of the collection.

Example

This example shows how to use the COLLECTION_MEMBER_COUNT function to get the total number of members in the DEPARTMENTS collection.

```
Begin
    l_count := APEX_COLLECTION.COLLECTION_MEMBER_COUNT( p_collection_name =>
'DEPARTMENTS';
End;
```

CREATE_COLLECTION Procedure

Use this procedure to create an empty collection that does not already exist. If a collection exists with the same name for the current user in the same session for the current Application ID, an application error is raised.

Syntax

```
APEX_COLLECTION.CREATE_COLLECTION(
    p_collection_name IN VARCHAR2);
```

Parameters

Table 12–7 describes the parameters available in the CREATE_COLLECTION procedure.

Table 12–7 CREATE_COLLECTION Procedure Parameters

Parameter	Description
p_collection_name	The name of the collection. The maximum length is 255 characters. An error is returned if this collection exists with the specified name of the current user and in the same session.

Example

This example shows how to use the CREATE_COLLECTION procedure to create an empty collection named EMPLOYEES.

```
Begin
    APEX_COLLECTION.CREATE_COLLECTION(
        p_collection_name => 'EMPLOYEES');
End;
```

See Also: "CREATE_OR_TRUNCATE_COLLECTION Procedure" on page 12-22, "CREATE_COLLECTION_FROM_QUERY Procedure" on page 12-23, "CREATE_COLLECTION_FROM_QUERY2 Procedure" on page 12-25, "CREATE_COLLECTION_FROM_QUERY_B Procedure" on page 12-27, "CREATE_COLLECTION_FROM_QUERYB2 Procedure" on page 12-29

CREATE_OR_TRUNCATE_COLLECTION Procedure

Use this procedure to create a collection. If a collection exists with the same name for the current user in the same session for the current Application ID, all members of the collection are removed. In other words, the named collection is truncated.

Syntax

```
APEX_COLLECTION.CREATE_OR_TRUNCATE_COLLECTION(
    p_collection_name IN VARCHAR2);
```

Parameters

Table 12–8 describes the parameters available in the CREATE_OR_TRUNCATE_COLLECTION procedure.

Table 12–8 CREATE_OR_TRUNCATE_COLLECTION Procedure Parameters

Parameter	Description
p_collection_name	The name of the collection. The maximum length is 255 characters. All members of the named collection are removed if the named collection exists for the current user in the current session.

Example

This example shows how to use the CREATE_OR_TRUNCATE_COLLECTION procedure to remove all members in an existing collection named EMPLOYEES.

```
Begin
    APEX_COLLECTION.CREATE_OR_TRUNCATE_COLLECTION(
        p_collection_name => 'EMPLOYEES');
End;
```

See Also: "CREATE_COLLECTION Procedure" on page 12-21, "CREATE_COLLECTION_FROM_QUERY Procedure" on page 12-23, "CREATE_COLLECTION_FROM_QUERY2 Procedure" on page 12-25, "CREATE_COLLECTION_FROM_QUERY_B Procedure" on page 12-27, "CREATE_COLLECTION_FROM_QUERYB2 Procedure" on page 12-29

CREATE_COLLECTION_FROM_QUERY Procedure

Use this procedure to create a collection from a supplied query. The query will be parsed as the application owner. This method can be used with a query with up to 50 columns in the SELECT clause. These columns in the SELECT clause will populate the 50 character attributes of the collection (C001 through C050). If a collection exists with the same name for the current user in the same session for the current Application ID, an application error is raised.

Syntax

```
APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY (
    p_collection_name IN VARCHAR2,
    p_query IN VARCHAR2,
    p_generate_md5 IN VARCHAR2 default 'NO');
```

Parameters

Table 12–9 describes the parameters available in the CREATE_COLLECTION_FROM_QUERY procedure.

Table 12–9 CREATE_COLLECTION_FROM_QUERY Procedure Parameters

Parameter	Description
p_collection_name	The name of the collection. The maximum length is 255 characters. An error is returned if this collection exists with the specified name of the current user and in the same session.
p_query	Query to execute in order to populate the members of the collection. If p_query is numeric, it is assumed to be a DBMS_SQL cursor.
p_generate_md5	Valid values include YES and NO. YES to specify if the message digest of the data of the collection member should be computed. Use this parameter to compare the MD5 of the collection member with another member or to see if that member has changed.

Example

The following example shows how to use the CREATE_COLLECTION_FROM_QUERY procedure to create a collection named AUTO and populate it with data from the AUTOS table. Because p_generate_md5 is 'YES', the MD5 checksum will be computed to allow comparisons to determine change status.

```
Begin
    l_query := 'select make, model, year from AUTOS';
    APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY (
        p_collection_name => 'AUTO',
        p_query => l_query,
        p_generate_md5 => 'YES');
End;
```

See Also: "GET_MEMBER_MD5 Function" on page 12-36, "CREATE_COLLECTION Procedure" on page 12-21, "CREATE_OR_TRUNCATE_COLLECTION Procedure" on page 12-22, "CREATE_COLLECTION_FROM_QUERY2 Procedure" on page 12-25, "CREATE_COLLECTION_FROM_QUERY_B Procedure" on page 12-27, "CREATE_COLLECTION_FROM_QUERYB2 Procedure" on page 12-29

CREATE_COLLECTION_FROM_QUERY2 Procedure

Use this procedure to create a collection from a supplied query. This method is identical to CREATE_COLLECTION_FROM_QUERY, however, the first 5 columns of the SELECT clause must be numeric and the next 5 must be date. After the numeric and date columns, there can be up to 50 character columns in the SELECT clause. The query will be parsed as the application owner. If a collection exists with the same name for the current user in the same session for the current Application ID, an application error is raised.

Syntax

```
APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY2 (
    p_collection_name IN VARCHAR2,
    p_query IN VARCHAR2,
    p_generate_md5 IN VARCHAR2 default 'NO');
```

Parameters

Table 12–10 describes the parameters available in the CREATE_COLLECTION_FROM_QUERY2 procedure.

Table 12–10 CREATE_COLLECTION_FROM_QUERY2 Procedure Parameters

Parameter	Description
p_collection_name	The name of the collection. The maximum length is 255 characters. An error is returned if this collection exists with the specified name of the current user and in the same session.
p_query	Query to executed in order to populate the members of the collection. If p_query is numeric, it is assumed to be a DBMS_SQL cursor.
p_generate_md5	Valid values include YES and NO. YES to specify if the message digest of the data of the collection member should be computed. Use this parameter to compare the MD5 of the collection member with another member or to see if that member has changed.

Example

The following example shows how to use the CREATE_COLLECTION_FROM_QUERY2 procedure to create a collection named EMPLOYEE and populate it with data from the EMP table. The first five columns (mgr, sal, comm, deptno, and null) are all numeric. Because p_generate_md5 is 'NO', the MD5 checksum is not computed.

```
begin;
    APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY2 (
        p_collection_name => 'EMPLOYEE',
        p_query => 'select empno, sal, comm, deptno, null, hiredate, null, null,
null, null, ename, job, mgr from emp',
        p_generate_md5 => 'NO');
end;
```

See Also: "GET_MEMBER_MD5 Function" on page 12-36, "CREATE_COLLECTION Procedure" on page 12-21, "CREATE_OR_TRUNCATE_COLLECTION Procedure" on page 12-22, "CREATE_COLLECTION_FROM_QUERY Procedure" on page 12-23, "CREATE_COLLECTION_FROM_QUERY_B Procedure" on page 12-27, "CREATE_COLLECTION_FROM_QUERYB2 Procedure" on page 12-29

CREATE_COLLECTION_FROM_QUERY_B Procedure

Use this procedure to create a collection from a supplied query using bulk operations. This method offers significantly faster performance than the CREATE_COLLECTION_FROM_QUERY method. The query will be parsed as the application owner. If a collection exists with the same name for the current user in the same session for the current Application ID, an application error is raised.

This procedure uses bulk dynamic SQL to perform the fetch and insert operations into the named collection. Two limitations are imposed by this procedure:

1. The MD5 checksum for the member data will not be computed.
2. No column value in query p_query can exceed 2,000 bytes. If a row is encountered that has a column value of more than 2,000 bytes, an error will be raised during execution. In Oracle Database 11gR2 11.2.0.1 or later, this column limit is 4,000 bytes.

Syntax

```
APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY_B (
    p_collection_name IN VARCHAR2,
    p_query IN VARCHAR2.
    p_names IN apex_application_global.vc_arr2 DEFAULT,
    p_values IN apex_applicatioN_globa.vc_arr2 DEFAULT,
    p_max_row_count IN NUMBER DEFAULT);
```

Parameters

Table 12–11 describes the parameters available in the CREATE_COLLECTION_FROM_QUERY_B procedure.

Table 12–11 CREATE_COLLECTION_FROM_QUERY_B Procedure Parameters

Parameter	Description
p_collection_name	The name of the collection. The maximum length is 255 characters. An error is returned if this collection exists with the specified name of the current user and in the same session.
p_query	Query to executed in order to populate the members of the collection. If p_query is numeric, it is assumed to be a DBMS_SQL cursor.
p_names	Array of bind variable names used in the query statement.
p_values	Array of bind variable values used in the bind variables in the query statement.
p_max_row_count	Maximum number of rows returned from the query in p_query which should be added to the collection.

Example

The following examples shows how to use the CREATE_COLLECTION_FROM_QUERY_B procedure to create a collection named AUTO and populate it with data from the AUTOS table.

```
Begin
    l_query := 'select make, model, year from AUTOS';
    APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY_B (
        p_collection_name => 'AUTO',
        p_query => l_query);
```

```
End;
```

See Also: "GET_MEMBER_MD5 Function" on page 12-36, "CREATE_COLLECTION Procedure" on page 12-21, "CREATE_OR_TRUNCATE_COLLECTION Procedure" on page 12-22, "CREATE_COLLECTION_FROM_QUERY Procedure" on page 12-23, "CREATE_COLLECTION_FROM_QUERY2 Procedure" on page 12-25, "CREATE_COLLECTION_FROM_QUERYB2 Procedure" on page 12-29

CREATE_COLLECTION_FROM_QUERYB2 Procedure

Use this procedure to create a collection from a supplied query using bulk operations. This method offers significantly faster performance than the CREATE_COLLECTION_FROM_QUERY_2 method. The query will be parsed as the application owner. If a collection exists with the same name for the current user in the same session for the current Application ID, an application error is raised. It is identical to the CREATE_COLLECTION_FROM_QUERY_B, however, the first five columns of the SELECT clause must be numeric and the next five columns must be date. After the date columns, there can be up to 50 character columns in the SELECT clause

This procedure uses bulk dynamic SQL to perform the fetch and insert operations into the named collection. Two limitations are imposed by this procedure:

1. The MD5 checksum for the member data will not be computed.
2. No column value in query p_query can exceed 2,000 bytes. If a row is encountered that has a column value of more than 2,000 bytes, an error will be raised during execution. In Oracle Database 11gR2 11.2.0.1 or later, this column limit is 4,000 bytes.

Syntax

```
APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERYB2 (
    p_collection_name IN VARCHAR2,
    p_query IN VARCHAR2.
    p_names IN apex_application_global.vc_arr2 DEFAULT,
    p_values IN apex_applicatioN_globa.vc_arr2 DEFAULT,
    p_max_row_count IN NUMBER DEFAULT);
```

Parameters

Table 12–12 describes the parameters available in the CREATE_COLLECTION_FROM_QUERYB2 procedure.

Table 12–12 CREATE_COLLECTION_FROM_QUERYB2 Procedure Parameters

Parameter	Description
p_collection_name	The name of the collection. The maximum length is 255 characters. An error is returned if this collection exists with the specified name of the current user and in the same session.
p_query	Query to executed in order to populate the members of the collection. If p_query is numeric, it is assumed to be a DBMS_SQL cursor.
p_names	Array of bind variable names used in the query statement.
p_values	Array of bind variable values used in the bind variables in the query statement.
p_max_row_count	Maximum number of rows returned from the query in p_query which should be added to the collection.

Example

The following example shows how to use the CREATE_COLLECTION_FROM_QUERYB2 procedure to create a collection named EMPLOYEES and populate it with data from the EMP table. The first five columns (mgr, sal, comm, deptno, and null) are all numeric and the next five are all date. Because p_generate_md5 is 'NO', the MD5 checksum is not computed.

```
Begin
    l_query := 'select empno, sal, comm, deptno, null, hiredate, null, null, null,
null, ename, job, mgr from emp';
    APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERYB2 (
        p_collection_name => 'EMPLOYEES',
        p_query => l_query,
        p_generate_md5 => 'NO');
End;
```

See Also: "GET_MEMBER_MD5 Function" on page 12-36, "CREATE_COLLECTION Procedure" on page 12-21, "CREATE_OR_TRUNCATE_COLLECTION Procedure" on page 12-22, "CREATE_COLLECTION_FROM_QUERY Procedure" on page 12-23, "CREATE_COLLECTION_FROM_QUERY2 Procedure" on page 12-25, "CREATE_COLLECTION_FROM_QUERY_B Procedure" on page 12-27

DELETE_ALL_COLLECTIONS Procedure

Use this procedure to delete all collections that belong to the current user in the current Application Express session regardless of the current Application ID.

Syntax

```
APEX_COLLECTION.DELETE_ALL_COLLECTIONS;
```

Parameters

None.

Example

This example shows how to use the `DELETE_ALL_COLLECTIONS` procedure to remove all collections that belong to the current user in the current session and Application ID.

```
Begin
    APEX_COLLECTION.DELETE_ALL_COLLECTIONS;
End;
```

> **See Also:** "DELETE_ALL_COLLECTIONS Procedure" on page 12-31, "DELETE_COLLECTION Procedure" on page 12-33, "DELETE_MEMBER Procedure" on page 12-34, "DELETE_MEMBERS Procedure" on page 12-35

DELETE_ALL_COLLECTIONS_SESSION Procedure

Use this procedure to delete all collections that belong to the current user in the current Flow session regardless of the Application ID.

Syntax

```
APEX_COLLECTION.DELETE_ALL_COLLECTIONS_SESSION;
```

Parameters

None.

Example

This example shows how to use the `DELETE_ALL_COLLECTIONS_SESSION` procedure to remove all collections that belong to the current user in the current session regardless of Application ID.

```
Begin
    APEX_COLLECTION.DELETE_ALL_COLLECTIONS_SESSION;
End;
```

> **See Also:** "DELETE_ALL_COLLECTIONS Procedure" on page 12-31, "DELETE_COLLECTION Procedure" on page 12-33, "DELETE_MEMBER Procedure" on page 12-34, "DELETE_MEMBERS Procedure" on page 12-35

DELETE_COLLECTION Procedure

Use this procedure to delete a named collection. All members that belong to the collection are removed and the named collection is dropped. If the named collection does not exist for the same user in the current session for the current Application ID, an application error is raised.

Syntax

```
APEX_COLLECTION.DELETE_COLLECTION (
    p_collection_name IN VARCHAR2);
```

Parameters

Table 12–13 describes the parameters available in the DELETE_COLLECTION procedure.

Table 12–13 DELETE_COLLECTION Procedure Parameters

Parameter	Description
p_collection_name	The name of the collection to remove all members from and drop. An error is returned if this collection does not exist with the specified name of the current user and in the same session.

Example

This example shows how to use the DELETE_COLLECTION procedure to remove the 'EMPLOYEE' collection.

```
Begin
    APEX_COLLECTION.DELETE_COLLECTION(
        p_collection_name => 'EMPLOYEE');
End;
```

See Also: "DELETE_ALL_COLLECTIONS_SESSION Procedure" on page 12-32, "DELETE_ALL_COLLECTIONS Procedure" on page 12-31, "DELETE_MEMBER Procedure" on page 12-34, "DELETE_MEMBERS Procedure" on page 12-35

DELETE_MEMBER Procedure

Use this procedure to delete a specified member from a given named collection. If the named collection does not exist for the same user in the current session for the current Application ID, an application error is raised.

Syntax

```
APEX_COLLECTION.DELETE_MEMBER (
    p_collection_name IN VARCHAR2,
    p_seq IN VARCHAR2);
```

Parameters

Table 12–14 describes the parameters available in the DELETE_MEMBER procedure.

Table 12–14 DELETE_MEMBER Parameters

Parameter	Description
p_collection_name	The name of the collection to delete the specified member from. The maximum length is 255 characters. Collection names are not case sensitive and are converted to upper case. An error is returned if this collection does not exist for the current user in the same session.
p_seq	This is the sequence ID of the collection member to be deleted.

Example

This example shows how to use the DELETE_MEMBER procedure to remove the member with a sequence ID of '2' from the collection named EMPLOYEES.

```
Begin
    APEX_COLLECTION.DELETE_MEMBER(
        p_collection_name => 'EMPLOYEES',
        p_seq => '2');
End;
```

See Also: "DELETE_ALL_COLLECTIONS_SESSION Procedure" on page 12-32, "DELETE_ALL_COLLECTIONS Procedure" on page 12-31, "DELETE_COLLECTION Procedure" on page 12-33, "DELETE_MEMBERS Procedure" on page 12-35

DELETE_MEMBERS Procedure

Use this procedure to delete all members from a given named collection where the attribute specified by the attribute number equals the supplied value. If the named collection does not exist for the same user in the current session for the current Application ID, an application error is raised. If the attribute number specified is invalid or outside the range of 1 to 50, an error is raised.

If the supplied attribute value is null, then all members of the named collection are deleted where the attribute, specified by `p_attr_number`, is null.

Syntax

```
APEX_COLLECTION.DELETE_MEMBERS (
    p_collection_name IN VARCHAR2,
    p_attr_number IN VARCHAR2,
    p_attr_value IN VARCHAR2);
```

Parameters

Table 12–14 describes the parameters available in the `DELETE_MEMBERS` procedure.

Table 12–15 DELETE_MEMBERS Parameters

Parameter	Description
`p_collection_name`	The name of the collection to delete the specified members from. The maximum length is 255 characters. Collection names are not case sensitive and are converted to upper case. An error is returned if this collection does not exist for the current user in the same session.
`p_attr_number`	Attribute number of the member attribute used to match for the specified attribute value for deletion. Valid values are 1 through 50 and null.
`p_attr_value`	Attribute value of the member attribute used to match for deletion. Maximum length can be 4,000 bytes. The attribute value will be truncated to 4,000 bytes if greater than this amount.

Example

The following example deletes all members of the collection named 'GROCERIES' where the 5th character attribute is equal to 'APPLE'.

```
Begin
    apex_collection.delete_members(
        p_collection_name => 'GROCERIES'
        p_attr_number     => 5,
        p_attr_value      => 'APPLE' );
    Commit;
End;
```

See Also: "DELETE_ALL_COLLECTIONS_SESSION Procedure" on page 12-32, "DELETE_ALL_COLLECTIONS Procedure" on page 12-31, "DELETE_COLLECTION Procedure" on page 12-33, "DELETE_MEMBER Procedure" on page 12-34

GET_MEMBER_MD5 Function

This function is used to compute and return the message digest of the attributes for the member specified by the sequence ID. This computation of message digest is equal to the computation performed natively by collections. Thus, the result of this function could be compared to the MD5_ORIGINAL column of the view wwv_flow_collections.

If a collection does not exist with the specified name for the current user in the same session and for the current Application ID, an application error will be raised. If the member specified by sequence ID p_seq does not exist, an application error will be raised.

Syntax

```
APEX_COLLECTION.GET_MEMBER_MD5 (
    p_collection_name IN VARCHAR2,
    p_seq IN NUMBER)
RETURN VARCHAR2;
```

Parameters

Table 12–16 describes the parameters available in the GET_MEMBER_MD5 function.

Table 12–16 GET_MEMBER_MD5 Parameters

Parameter	Description
p_collection_name	The name of the collection to add this array of members to. An error is returned if this collection does not exist with the specified name of the current user and in the same session.
p_seq	Sequence ID of the collection member.

Example

The following example computes the MD5 for the 5th member of the GROCERIES collection.

```
Begin
    apex_collection.delete_members(
        p_collection_name => 'GROCERIES'
        p_attr_number     => 5,
        p_attr_value      => 'APPLE' );
    Commit;
End;
```

See Also: "COLLECTION_HAS_CHANGED Function" on page 12-19, "RESET_COLLECTION_CHANGED Procedure" on page 12-42, "RESET_COLLECTION_CHANGED_ALL Procedure" on page 12-43

MERGE_MEMBERS Procedure

Use this procedure to merge members of the given named collection with the values passed in the arrays. If the named collection does not exist one is created. If a p_init_query is provided, the collection is created from the supplied SQL query. If the named collection exists, the following occurs:

1. Rows in the collection and not in the arrays are deleted.
2. Rows in the collections and in the arrays are updated.
3. Rows in the arrays and not in the collection are inserted.

The count of elements in the p_c001 PL/SQL table will be used as the total number of items across all PL/SQL tables. For example, if p_c001.count is 2 and p_c002.count is 10, only 2 members will be merged. If p_c001 is null an application error is raised.

Syntax

```
APEX_COLLECTION.MERGE_MEMBERS (
    p_collection_name IN VARCHAR2,
    p_seq  IN APEX_APPLICATION_GLOBAL.VC_ARR2 DEFAULT empty_vc_arr,
    p_c001 IN APEX_APPLICATION_GLOBAL.VC_ARR2 DEFAULT empty_vc_arr,
    p_c002 IN APEX_APPLICATION_GLOBAL.VC_ARR2 DEFAULT empty_vc_arr,
    p_c003 IN APEX_APPLICATION_GLOBAL.VC_ARR2 DEFAULT empty_vc_arr,
    ...
    p_c050 IN APEX_APPLICATION_GLOBAL.VC_ARR2 DEFAULT empty_vc_arr,
    p_null_index  IN NUMBER DEFAULT 1,
    p_null_value  IN VARCHAR2 DEFAULT null,
    p_init_query  IN VARCHAR2 DEFAULT null);
```

Parameters

Table 12–17 describes the parameters available in the `MERGE_MEMBERS` procedure.

> **Note:** Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. Also, the number of members added is based on the number of elements in the first array.

Table 12–17 MERGE_MEMBERS Procedure Parameters

Parameter	Description
`p_collection_name`	The name of the collection. Maximum length is 255 bytes. Collection names are not case sensitive and are converted to upper case.
`p_c001 through p_c050`	Array of attribute values to be merged. Maximum length is 4,000 bytes. Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. The count of the p_c001 array is used across all arrays. If no values are provided then no actions are performed.
`p_c0xx`	Attribute of NN attributes values to be merged. Maximum length can be 4,000 bytes. The attribute value is truncated to 4,000 bytes if greater than this amount.
`p_seq`	Identifies the sequence number of the collection to be merged.

Table 12–17 (Cont.) MERGE_MEMBERS Procedure Parameters

Parameter	Description
p_null_index	That is if the element identified by this value is null, then treat this row as a null row. For example, if p_null_index is 3, then p_c003 is treated as a null row. In other words, tell the merge function to ignore this row. This results in the null rows being removed from the collection. The null index works in conjunction with the null value. If the value of the p_cXXX argument is equal to the p_null_value then the row will be treated as null.
p_null_value	Used in conjunction with the p_null_index argument. Identifies the null value. If used, this value must not be null. A typical value for this argument is "0"
p_init_query	If the collection does not exist, the collection is created using this query.

Example

The following example creates a collection on the table of employees, and then merges the contents of the local arrays with the collection, updating the job of two employees.

```
DECLARE
    l_seq   APEX_APPLICATION_GLOBAL.VC_ARR2;
    l_c001  APEX_APPLICATION_GLOBAL.VC_ARR2;
    l_c002  APEX_APPLICATION_GLOBAL.VC_ARR2;
    l_c003  APEX_APPLICATION_GLOBAL.VC_ARR2;
BEGIN
    l_seq(1)  := 1;
    l_c001(1) := 7369;
    l_c002(1) := 'SMITH';
    l_c003(1) := 'MANAGER';
    l_seq(2)  := 2;
    l_c001(2) := 7499;
    l_c002(2) := 'ALLEN';
    l_c003(2) := 'CLERK';

    APEX_COLLECTION.MERGE_MEMBERS(
        p_collection_name => 'EMPLOYEES',
        p_seq => l_seq,
        p_c001 => l_c001,
        p_c002 => l_c002,
        p_c003 => l_c003,
        p_init_query => 'select empno, ename, job from emp order by empno');
END;
```

MOVE_MEMBER_DOWN Procedure

Use this procedure to adjust the sequence ID of specified member in the given named collection down by one (subtract one), swapping sequence ID with the one it is replacing. For example, 3 becomes 2 and 2 becomes 3. If a collection does not exist with the specified name for the current user in the same session and for the current Application ID, an application error will be raised. If the member specified by sequence ID p_seq does not exist, an application error is raised. If the member specified by sequence ID p_seq is the lowest sequence in the collection, an application error is NOT returned.

Syntax

```
APEX_COLLECTION.MOVE_MEMBER_DOWN (
    p_collection_name IN VARCHAR2,
    p_seq IN NUMBER);
```

Parameters

Table 12–19 describes the parameters available in the MOVE_MEMBER_DOWN procedure.

Table 12–18 MOVE_MEMBER_DOWN Parameters

Parameter	Description
p_collection_name	The name of the collection. Maximum length is 255 bytes. Collection names are not case sensitive and are converted to upper case. An error is returned if this collection does not exist with the specified name of the current user in the same session.
p_seq	Identifies the sequence number of the collection member to be moved down by one.

Example

This example shows how to a member of the EMPLOYEES collection down one position. After executing this example, sequence ID '5' becomes sequence ID '4' and sequence ID '4' becomes sequence ID '5'.

```
BEGIN;
    APEX_COLLECTION.MOVE_MEMBER_DOWN(
        p_collection_name => 'EMPLOYEES',
        p_seq => '5' );
END;
```

See Also: "MOVE_MEMBER_UP Procedure" on page 12-40

MOVE_MEMBER_UP Procedure

Use this procedure to adjust the sequence ID of specified member in the given named collection up by one (add one), swapping sequence ID with the one it is replacing. For example, 2 becomes 3 and 3 becomes 2. If a collection does not exist with the specified name for the current user in the same session and for the current Application ID, an application error will be raised. If the member specified by sequence ID p_seq does not exist, an application error is raised. If the member specified by sequence ID p_seq is the highest sequence in the collection, an application error is not returned.

Syntax

```
APEX_COLLECTION.MOVE_MEMBER_UP (
    p_collection_name IN VARCHAR2,
    p_seq IN NUMBER);
```

Parameters

Table 12–19 describes the parameters available in the MOVE_MEMBER_UP procedure.

Table 12–19 MOVE_MEMBER_UP Parameters

Parameter	Description
p_collection_name	The name of the collection. Maximum length is 255 bytes. Collection names are not case sensitive and are converted to upper case. An error is returned if this collection does not exist with the specified name of the current user in the same session.
p_seq	Identifies the sequence number of the collection member to be moved up by one.

Example

This example shows how to a member of the EMPLOYEES collection down one position. After executing this example, sequence ID '5' becomes sequence ID '6' and sequence ID '6' becomes sequence ID '5'.

```
BEGIN;
    APEX_COLLECTION.MOVE_MEMBER_UP(
        p_collection_name => 'EMPLOYEES',
        p_seq => '5' );
END;
```

See Also: "MOVE_MEMBER_DOWN Procedure" on page 12-39

RESEQUENCE_COLLECTION Procedure

For a named collection, use this procedure to update the seq_id value of each member so that no gaps exist in the sequencing. For example, a collection with the following set of sequence IDs (1,2,3,5,8,9) becomes (1,2,3,4,5,6). If a collection does not exist with the specified name for the current user in the same session and for the current Application ID, an application error will be raised.

Syntax

```
APEX_COLLECTION.RESEQUENCE_COLLECTION (
    p_collection_name IN VARCHAR2);
```

Parameters

Table 12–20 describes the parameters available in the RESEQUENCE_COLLECTION procedure.

Table 12–20 RESEQUENCE_COLLECTION Parameters

Parameter	Description
p_collection_name	The name of the collection to resequence. An error is returned if this collection does not exist with the specified name of the current user and in the same session.

Example

This example shows how to resequence the DEPARTMENTS collection to remove gaps in the sequence IDs.

```
BEGIN;
    APEX_COLLECTION.RESEQUENCE_COLLECTION (
        p_collection_name => 'DEPARTMENTS');
END;
```

See Also: "MOVE_MEMBER_DOWN Procedure" on page 12-39, "MOVE_MEMBER_UP Procedure" on page 12-40

RESET_COLLECTION_CHANGED Procedure

Use this procedure to reset the collection changed flag (mark as not changed) for a given collection. If a collection does not exist with the specified name for the current user in the same session and for the current Application ID, an application error will be raised.

Syntax

```
APEX_COLLECTION.RESET_COLLECTION_CHANGED (
    p_collection_name IN VARCHAR2);
```

Parameters

Table 12–21 describes the parameters available in the RESET_COLLECTION_CHANGED procedure.

Table 12–21 RESET_COLLECTION_CHANGED Parameters

Parameter	Description
p_collection_name	The name of the collection to reset the collection changed flag. An error is returned if this collection does not exist with the specified name of the current user and in the same session.

Example

This example shows how to reset the changed flag for the DEPARTMENTS collection.

```
BEGIN;
    APEX_COLLECTION.RESET_COLLECTION_CHANGED (
        p_collection_name => 'DEPARTMENTS');
END;
```

See Also: "RESET_COLLECTION_CHANGED_ALL Procedure" on page 12-43

RESET_COLLECTION_CHANGED_ALL Procedure

Use this procedure to reset the collection changed flag (mark as not changed) for all collections in the user's current session.

Syntax

```
APEX_COLLECTION.RESET_COLLECTION_CHANGED_ALL; (
```

Parameters

None.

Example

This example shows how to reset the changed flag for all collections in the user's current session.

```
BEGIN;
    APEX_COLLECTION.RESET_COLLECTION_CHANGED_ALL;
END;
```

See Also: "RESET_COLLECTION_CHANGED Procedure" on page 12-42.

SORT_MEMBERS Procedure

Use this procedure to reorder the members of a given collection by the column number specified by p_sort_on_column_number. This will sort the collection by a particular column/attribute in the collection and reassign the sequence IDs of each number such that no gaps exist. If a collection does not exist with the specified name for the current user in the same session and for the current Application ID, an application error will be raised.

Syntax

```
APEX_COLLECTION.SORT_MEMBERS (
    p_collection_name IN VARCHAR2,
    p_sort_on_column_number IN NUMBER);
```

Parameters

Table 12–22 describes the parameters available in the SORT_MEMBERS procedure.

Table 12–22 SORT_MEMBERS Parameters

Parameter	Description
p_collection_name	The name of the collection to sort. An error is returned if this collection does not exist with the specified name of the current user and in the same session.
p_sort_on_column_number	The column number used to sort the collection.

Example

In this example, column 2 of the DEPARTMENTS collection is the department location. The collection is reorder according to the department location.

```
BEGIN;
    APEX_COLLECTION.SORT_MEMBERS (
        p_collection_name => 'DEPARTMENTS',
        p_sort_on_column_number => '2';
END;
```

TRUNCATE_COLLECTION Procedure

Use this procedure to remove all members from a named collection. If a collection does not exist with the specified name for the current user in the same session and for the current Application ID, an application error will be raised.

Syntax

```
APEX_COLLECTION.TRUNCATE_COLLECTION (
    p_collection_name IN VARCHAR2);
```

Parameters

Table 12–23 describes the parameters available in the `TRUNCATE_COLLECTION` procedure.

Table 12–23 TRUNCATE_COLLECTION Parameters

Parameter	Description
`p_collection_name`	The name of the collection to truncate. An error is returned if this collection does not exist with the specified name of the current user and in the same session.

Example

This example shows how to remove all members from the DEPARTMENTS collection.

```
BEGIN;
    APEX_COLLECTION.TRUNCATE_COLLECTION(
        p_collection_name => 'DEPARTMENTS');
END;
```

See Also: "CREATE_OR_TRUNCATE_COLLECTION Procedure" on page 12-22

UPDATE_MEMBER Procedure

Use this procedure to update the specified member in the given named collection. If a collection does not exist with the specified name for the current user in the same session and for the current Application ID, an application error will be raised. If the member specified by sequence ID p_seq does not exist, an application error is raised.

Syntax

```
APEX_COLLECTION.UPDATE_MEMBER (
    p_collection_name IN VARCHAR2,
    p_seq  IN VARCHAR2 DEFAULT NULL,
    p_c001 IN VARCHAR2 DEFAULT NULL,
    p_c002 IN VARCHAR2 DEFAULT NULL,
    p_c003 IN VARCHAR2 DEFAULT NULL,
    ...
    p_c050 IN VARCHAR DEFAULT NULL,
    p_n001 IN NUMBER DEFAULT NULL,
    p_n002 IN NUMBER DEFAULT NULL,
    p_n003 IN NUMBER DEFAULT NULL,
    p_n004 IN NUMBER DEFAULT NULL,
    p_n005 IN NUMBER DEFAULT NULL,
    p_d001 IN DATE DEFAULT NULL,
    p_d002 IN DATE DEFAULT NULL,
    p_d003 IN DATE DEFAULT NULL,
    p_d004 IN DATE DEFAULT NULL,
    p_d005 IN DATE DEFAULT NULL,
    p_clob001 IN CLOB DEFAULT empty_clob(),
    p_blob IN BLOB DEFAULT empty-blob(),
    p_xmltype001 IN XMLTYPE DEFAULT NULL);
```

Parameters

Table 12–24 describes the parameters available in the `UPDATE_MEMBER` procedure.

> **Note:** Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. Also, the number of members added is based on the number of elements in the first array.

Table 12–24 UPDATE_MEMBER Parameters

Parameter	Description
`p_collection_name`	The name of the collection to update. Maximum length is 255 bytes. Collection names are not case sensitive and are converted to upper case.
`p_c001 through p_c050`	Attribute value of the member to be added. Maximum length is 4,000 bytes. Any character attribute exceeding 4,000 characters is truncated to 4,000 characters.
`p_n001 through p_n005`	Attribute value of the numeric attributes to be added or updated.
`p_d001` through `p_d005`	Attribute value of the date attributes to be added or updated.
`p_clob001`	Use p_clob001 for collection member attributes that exceed 4,000 characters.
`p_blob001`	Use p_blob001 for binary collection member attributes.

Table 12–24 (Cont.) UPDATE_MEMBER Parameters

Parameter	Description
p_xmltype001	Use p_xmltype001 to store well-formed XML.

Example

Update the second member of the collection named 'Departments', updating the first member attribute to 'Engineering' and the second member attribute to 'Sales'.

```
BEGIN;
    APEX_COLLECTION.UPDATE_MEMBER (
        p_collection_name => 'Departments',
        p_seq => '2',
        p_c001 => 'Engineering',
        p_c002 => 'Sales');
```

See Also: "UPDATE_MEMBERS Procedure" on page 12-48

UPDATE_MEMBERS Procedure

Use this procedure to update the array of members for the given named collection. If a collection does not exist with the specified name for the current user in the same session and for the current Application ID, an application error will be raised. The count of elements in the p_seq PL/SQL table will be used as the total number of items across all PL/SQL tables. That is, if p_seq.count = 2 and p_c001.count = 10, only 2 members will be updated. If p_seq is null, an application error is raised. If the member specified by sequence ID p_seq does not exist, an application error is raised.

Syntax

```
APEX_COLLECTION.UPDATE_MEMBERS (
    p_collection_name IN VARCHAR2,
    p_seq  IN wwv_flow_global.VC_ARR2 DEFAULT empty_vc_arr,
    p_c001 IN wwv_flow_global.VC_ARR2 DEFAULT empty_vc_arr,
    p_c002 IN wwv_flow_global.VC_ARR2 DEFAULT empty_vc_arr,
    p_c003 IN wwv_flow_global.VC_ARR2 DEFAULT empty_vc_arr,
    ...
    p_c050 IN wwv_flow_global.VC_ARR2 DEFAULT empty_vc_arr,
    p_n001 IN wwv_flow_global.N_ARR DEFAULT empty_n_arr,
    p_n002 IN wwv_flow_global.N_ARR DEFAULT empty_n_arr,
    p_n003 IN wwv_flow_global.N_ARR DEFAULT empty_n_arr,
    p_n004 IN wwv_flow_global.N_ARR DEFAULT empty_n_arr,
    p_n005 IN wwv_flow_global.N_ARR DEFAULT empty_n_arr,
    p_d001 IN wwv_flow_global.D_ARR DEFAULT empty_d_arr,
    p_d002 IN wwv_flow_global.D_ARR DEFAULT empty_d_arr,
    p_d003 IN wwv_flow_global.D_ARR DEFAULT empty_d_arr,
    p_d004 IN wwv_flow_global.D_ARR DEFAULT empty_d_arr,
    p_d005 IN wwv_flow_global.D_ARR DEFAULT empty_d_arr)
```

Parameters

Table 12–25 describes the parameters available in the `UPDATE_MEMBERS` procedure.

> **Note:** Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. Also, the number of members added is based on the number of elements in the first array.

Table 12–25 UPDATE_MEMBERS Parameters

Parameter	Description
`p_collection_name`	The name of the collection to update. Maximum length is 255 bytes. Collection names are not case sensitive and are converted to upper case.
`p_seq`	Array of member sequence IDs to be updated. The count of the p_seq array is used across all arrays.
`p_c001 through p_c050`	Array of attribute values to be updated.
`p_n001 through p_n005`	Attribute value of numeric
`p_d001` through `p_d005`	Array of date attribute values to be updated.

Example

```
DECLARE
```

```
    l_seq   apex_application_global.vc_arr2;
    l_carr  apex_application_global.vc_arr2;
    l_narr  apex_application_global.n_arr;
    l_darr  apex_application_global.d_arr;
BEGIN
    l_seq(1)  := 10;
    l_seq(2)  := 15;
    l_carr(1) := 'Apples';
    l_carr(2) := 'Grapes';
    l_narr(1) := 100;
    l_narr(2) := 150;
    l_darr(1) := sysdate;
    l_darr(2) := sysdate;

    APEX_COLLECTION.UPDATE_MEMBERS (
        p_collection_name => 'Groceries',
        p_seq  => l_seq,
        p_c001 => l_carr,
        p_n001 => l_narr,
        p_d001 => l_darr);
END;
```

See Also: "UPDATE_MEMBER Procedure" on page 12-46

UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1

Update the specified member attribute in the given named collection. If a collection does not exist with the specified name for the current user in the same session for the current Application ID, an application error will be raised. If the member specified by sequence ID p_seq does not exist, an application error will be raised. If the attribute number specified is invalid or outside the range 1-50, an error will be raised. Any attribute value exceeding 4,000 bytes will be truncated to 4,000 bytes.

Syntax

```
APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
    p_collection_name IN VARCHAR2,
    p_seq IN VARCHAR2,
    p_attr_number IN VARCHAR2,
    p_attr_value  IN VARCHAR2);
```

Parameters

Table 12–26 describes the parameters available in the `UPDATE_MEMBER_ATTRIBUTE` procedure signature 1.

> **Note:** Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. Also, the number of members added is based on the number of elements in the first array.

Table 12–26 UPDATE_MEMBER_ATTRIBUTE Signature 1 Parameters

Parameter	Description
`p_collection_name`	The name of the collection. Maximum length can be 255 bytes. Collection_names are case-insensitive, as the collection name will be converted to upper case. An error is returned if this collection does not exist with the specified name of the current user and in the same session.
`p_seq`	Sequence ID of the collection member to be updated.
`p_attr_number`	Attribute number of the member attribute to be updated. Valid values are 1 through 50. Any number outside of this range will be ignored.
`p_attr_value`	Attribute value of the member attribute to be updated.

Example

```
BEGIN;
    APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
        p_collection_name =. 'Departments',
        p_seq => '2',
        p_attr_number => '1',
        p_attr_value => 'Engineering');
END;
```

See Also: "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2" on page 12-52, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 3" on page 12-54, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4" on page 12-56, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 5" on page 12-58, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6" on page 12-60

UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2

Update the specified CLOB member attribute in the given named collection. If a collection does not exist with the specified name for the current user in the same session for the current Application ID, an application error will be raised. If the member specified by sequence ID p_seq does not exist, an application error will be raised. If the attribute number specified is invalid or outside the valid range (currently only 1 for CLOB), an error will be raised.

Syntax

```
APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
    p_collection_name IN VARCHAR2,
    p_seq IN VARCHAR2,
    p_clob_number IN NUMBER,
    p_clob_value  IN CLOB);
```

Parameters

Table 12–27 describes the parameters available in the `UPDATE_MEMBER_ATTRIBUTE` procedure signature 2.

> **Note:** Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. Also, the number of members added is based on the number of elements in the first array.

Table 12–27 UPDATE_MEMBER_ATTRIBUTE Signature 2 Parameters

Parameter	Description
`p_collection_name`	The name of the collection. Maximum length can be 255 bytes. Collection_names are case-insensitive, as the collection name will be converted to upper case. An error is returned if this collection does not exist with the specified name of the current user and in the same session.
`p_seq`	Sequence ID of the collection member to be updated.
`p_clob_number`	Attribute number of the CLOB member attribute to be updated. Valid value is 1. Any number outside of this range will be ignored.
`p_clob_value`	Attribute value of the CLOB member attribute to be updated.

Example

The following example sets the first and only CLOB attribute of collection sequence number 2 in the collection named 'Departments' to a value of 'Engineering'.

```
BEGIN;
    APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
        p_collection_name =. 'Departments',
        p_seq => '2',
        p_clob_number => '1',
        p_clob_value => 'Engineering');
END;
```

See Also: "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1" on page 12-50, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 3" on page 12-54, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4" on page 12-56, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 5" on page 12-58, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6" on page 12-60

UPDATE_MEMBER_ATTRIBUTE Procedure Signature 3

Update the specified BLOB member attribute in the given named collection. If a collection does not exist with the specified name for the current user in the same session for the current Application ID, an application error will be raised. If the member specified by sequence ID p_seq does not exist, an application error will be raised. If the attribute number specified is invalid or outside the valid range (currently only 1 for BLOB), an error will be raised.

Syntax

```
APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
    p_collection_name IN VARCHAR2,
    p_seq IN VARCHAR2,
    p_blob_number IN NUMBER,
    p_blob_value  IN BLOB);
```

Parameters

Table 12–28 describes the parameters available in the `UPDATE_MEMBER_ATTRIBUTE` procedure signature 3.

> **Note:** Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. Also, the number of members added is based on the number of elements in the first array.

Table 12–28 UPDATE_MEMBER_ATTRIBUTE Signature 3 Parameters

Parameter	Description
`p_collection_name`	The name of the collection. Maximum length can be 255 bytes. Collection_names are case-insensitive, as the collection name will be converted to upper case. An error is returned if this collection does not exist with the specified name of the current user and in the same session.
`p_seq`	Sequence ID of the collection member to be updated.
`p_blob_number`	Attribute number of the BLOB member attribute to be updated. Valid value is 1. Any number outside of this range will be ignored.
`p_blob_value`	Attribute value of the BLOB member attribute to be updated.

Example

The following example sets the first and only BLOB attribute of collection sequence number 2 in the collection named 'Departments' to a value of the BLOB variable `l_blob_content`.

```
BEGIN;
    APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
        p_collection_name =. 'Departments',
        p_seq => '2',
        p_blob_number => '1',
        p_blob_value => l_blob_content);
END;
```

See Also: "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1" on page 12-50, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2" on page 12-52, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4" on page 12-56, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 5" on page 12-58, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6" on page 12-60

UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4

Update the specified XMLTYPE member attribute in the given named collection. If a collection does not exist with the specified name for the current user in the same session for the current Application ID, an application error will be raised. If the member specified by sequence ID p_seq does not exist, an application error will be raised. If the attribute number specified is invalid or outside the valid range (currently only 1 for XMLTYPE), an error will be raised.

Syntax

```
APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
    p_collection_name IN VARCHAR2,
    p_seq IN VARCHAR2,
    p_xmltype_number IN NUMBER,
    p_xmltype_value  IN BLOB);
```

Parameters

Table 12–29 describes the parameters available in the `UPDATE_MEMBER_ATTRIBUTE` procedure signature 4.

> **Note:** Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. Also, the number of members added is based on the number of elements in the first array.

Table 12–29 UPDATE_MEMBER_ATTRIBUTE Signature 4 Parameters

Parameter	Description
`p_collection_name`	The name of the collection. Maximum length can be 255 bytes. Collection_names are case-insensitive, as the collection name will be converted to upper case. An error is returned if this collection does not exist with the specified name of the current user and in the same session.
`p_seq`	Sequence ID of the collection member to be updated.
`p_xmltype_number`	Attribute number of the XMLTYPE member attribute to be updated. Valid value is 1. Any number outside of this range will be ignored.
`p_xmltype_value`	Attribute value of the XMLTYPE member attribute to be updated.

Example

The following example sets the first and only XML attribute of collection sequence number 2 in the collection named 'Departments' to a value of the XMLType variable `l_xmltype_content`.

```
BEGIN;
    APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
        p_collection_name =. 'Departments',
        p_seq => '2',
        p_xmltype_number => '1',
        p_xmltype_value => l_xmltype_content);
END;
```

See Also: "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1" on page 12-50, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2" on page 12-52, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 3" on page 12-54, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 5" on page 12-58, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6" on page 12-60

UPDATE_MEMBER_ATTRIBUTE Procedure Signature 5

Update the specified NUMBER member attribute in the given named collection. If a collection does not exist with the specified name for the current user in the same session for the current Application ID, an application error will be raised. If the member specified by sequence ID p_seq does not exist, an application error will be raised. If the attribute number specified is invalid or outside the valid range (currently only 1 through 5 for NUMBER), an error will be raised.

Syntax

```
APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
    p_collection_name IN VARCHAR2,
    p_seq IN VARCHAR2,
    p_attr_number IN NUMBER,
    p_number_value  IN NUMBER);
```

Parameters

Table 12–30 describes the parameters available in the `UPDATE_MEMBER_ATTRIBUTE` procedure signature 5.

> **Note:** Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. Also, the number of members added is based on the number of elements in the first array.

Table 12–30 UPDATE_MEMBER_ATTRIBUTE Signature 5 Parameters

Parameter	Description
`p_collection_name`	The name of the collection. Maximum length can be 255 bytes. Collection_names are case-insensitive, as the collection name will be converted to upper case. An error is returned if this collection does not exist with the specified name of the current user and in the same session.
`p_seq`	Sequence ID of the collection member to be updated.
`p_attr_number`	Attribute number of the NUMBER member attribute to be updated. Valid value is 1 through 5. Any number outside of this range will be ignored.
`p_number_value`	Attribute value of the NUMBER member attribute to be updated.

Example

The following example sets the first numeric attribute of collection sequence number 2 in the collection named 'Departments' to a value of 3000.

```
BEGIN;
    APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
        p_collection_name =. 'Departments',
        p_seq => '2',
        p_attr_number => '1',
        p_number_value => 3000);
END;
```

See Also: "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1" on page 12-50, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2" on page 12-52, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 3" on page 12-54, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4" on page 12-56, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6" on page 12-60, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6" on page 12-60

UPDATE_MEMBER_ATTRIBUTE Procedure Signature 6

Update the specified DATE member attribute in the given named collection. If a collection does not exist with the specified name for the current user in the same session for the current Application ID, an application error will be raised. If the member specified by sequence ID p_seq does not exist, an application error will be raised. If the attribute number specified is invalid or outside the valid range (currently only 1 through 5 for DATE), an error will be raised.

Syntax

```
APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
    p_collection_name IN VARCHAR2,
    p_seq IN VARCHAR2,
    p_attr_number IN NUMBER,
    p_number_value  IN DATE);
```

Parameters

Table 12–30 describes the parameters available in the `UPDATE_MEMBER_ATTRIBUTE` procedure signature 6.

> **Note:** Any character attribute exceeding 4,000 characters is truncated to 4,000 characters. Also, the number of members added is based on the number of elements in the first array.

Table 12–31 UPDATE_MEMBER_ATTRIBUTE Signature 6 Parameters

Parameter	Description
`p_collection_name`	The name of the collection. Maximum length can be 255 bytes. Collection_names are case-insensitive, as the collection name will be converted to upper case. An error is returned if this collection does not exist with the specified name of the current user and in the same session.
`p_seq`	Sequence ID of the collection member to be updated.
`p_attr_number`	Attribute number of the DATE member attribute to be updated. Valid value is 1 through 5. Any number outside of this range will be ignored.
`p_number_value`	Attribute value of the DATE member attribute to be updated.

Example

Update the first attribute of the second collection member in collection named 'Departments', and set it to a value of 100.

```
BEGIN;
    APEX_COLLECTION.UPDATE_MEMBER_ATTRIBUTE (
        p_collection_name =. 'Departments',
        p_seq => '2',
        p_attr_number => '1',
        p_number_value => 100 );
END;
```

See Also: "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 1" on page 12-50, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 2" on page 12-52, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 3" on page 12-54, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 4" on page 12-56, "UPDATE_MEMBER_ATTRIBUTE Procedure Signature 5" on page 12-58

13

APEX_WEB_SERVICE

The APEX_WEB_SERVICE API allows you to integrate other systems with Application Express by allowing you to interact with Web services anywhere you can utilize PL/SQL in your application. The API contains procedures and functions to call both SOAP and RESTful style Web services. It contains functions to parse the responses from Web services and to encode/decode into SOAP friendly base64 encoding.

This API also contains package globals for managing cookies and HTTP headers when calling Web services whether from the API or by using standard processes of type Web service. Cookies and HTTP headers can be set prior to invoking a call to a Web service by populating the globals and the cookies and HTTP headers returned from the Web service response can be read from other globals.

Topics:

- About the APEX_WEB_SERVICE API
- BLOB2CLOBBASE64 Function
- CLOBBASE642BLOB Function
- MAKE_REQUEST Procedure
- MAKE_REQUEST Function
- MAKE_REST_REQUEST Function
- PARSE_RESPONSE Function
- PARSE_RESPONSE_CLOB Function
- PARSE_XML Function
- PARSE_XML_CLOB Function

About the APEX_WEB_SERVICE API

Use the APEX_WEB_SERVICE API to invoke a Web service and examine the response anywhere you can utilize PL/SQL in Application Express.

The following are examples of when you might use the APEX_WEB_SERVICE API:

- When you want to invoke a Web service via an On Demand Process using AJAX.
- When you want to invoke a Web service as part of an Authentication Scheme.
- When you need to pass a large binary parameter to a Web service that is base64 encoded.
- When you want to invoke a Web service as part of a validation.

Topics:

- Invoking a SOAP Style Web Service
- Invoking a RESTful Style Web Service
- Retrieving Cookies and HTTP Headers
- Setting Cookies and HTTP Headers

Invoking a SOAP Style Web Service

There is a procedure and a function to invoke a SOAP style Web service. The procedure will store the response in the collection specified by the parameter p_collection_name. The function will return the results as an XMLTYPE. To retrieve a specific value from the response, you use either the PARSE_RESPONSE function if the result is stored in a collection or the PARSE_XML function if the response is returned as an XMLTYPE.

If you need to pass a binary parameter to the Web service as base64 encoded character data, use the function BLOB2CLOBBASE64. Conversely, if you need to transform a response that contains a binary parameter that is base64 encoded use the function CLOBBASE642BLOB.

The following is an example of using the BLOB2CLOBBASE64 function to encode a parameter, MAKE_REQUEST procedure to call a Web service, and the PARSE_RESPONSE function to extract a specific value from the response.

```
declare
 l_filename varchar2(255);
 l_BLOB BLOB;
 l_CLOB CLOB;
 l_envelope CLOB;
 l_response_msg varchar2(32767);
BEGIN
 IF :P1_FILE IS NOT NULL THEN
    SELECT filename, BLOB_CONTENT
      INTO l_filename, l_BLOB
      FROM APEX_APPLICATION_FILES
      WHERE name = :P1_FILE;

    l_CLOB := apex_web_service.blob2clobbase64(l_BLOB);

   l_envelope := q'!<?xml version='1.0' encoding='UTF-8'?>!';
   l_envelope := l_envelope '<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:chec="http://www.stellent.com/CheckIn/">
  <soapenv:Header/>
  <soapenv:Body>
     <chec:CheckInUniversal>
        <chec:dDocName>'||l_filename||'</chec:dDocName>
        <chec:dDocTitle>'||l_filename||'</chec:dDocTitle>
        <chec:dDocType>Document</chec:dDocType>
        <chec:dDocAuthor>GM</chec:dDocAuthor>
        <chec:dSecurityGroup>Public</chec:dSecurityGroup>
        <chec:dDocAccount></chec:dDocAccount>
        <chec:CustomDocMetaData>
           <chec:property>
              <chec:name></chec:name>
              <chec:value></chec:value>
           </chec:property>
        </chec:CustomDocMetaData>
        <chec:primaryFile>
           <chec:fileName>'||l_filename'||</chec:fileName>
           <chec:fileContent>'||l_CLOB||'</chec:fileContent>
        </chec:primaryFile>
        <chec:alternateFile>
           <chec:fileName></chec:fileName>
           <chec:fileContent></chec:fileContent>
```

```
        </chec:alternateFile>
        <chec:extraProps>
           <chec:property>
              <chec:name></chec:name>
              <chec:value></chec:value>
           </chec:property>
        </chec:extraProps>
     </chec:CheckInUniversal>
  </soapenv:Body>
</soapenv:Envelope>';

apex_web_service.make_request(
   p_url               => 'http://127.0.0.1/idc/idcplg',
   p_action            => 'http://www.stellent.com/CheckIn/',
   p_collection_name   => 'STELLENT_CHECKIN',
   p_envelope          => l_envelope,
   p_username          => 'sysadmin',
   p_password          => 'welcome1' );

 l_response_msg := apex_web_service.parse_response(
  p_collection_name=>'STELLENT_CHECKIN',
p_
xpath=>'//idc:CheckInUniversalResponse/idc:CheckInUniversalResult/idc:StatusInfo/i
dc:statusMessage/text()',
  p_ns=>'xmlns:idc="http://www.stellent.com/CheckIn/"');

 :P1_RES_MSG := l_response_msg;

 END IF;
END;
```

Invoking a RESTful Style Web Service

RESTful style Web services use a simpler architecture than SOAP. Typically the input to a RESTful style Web service is a collection of name/value pairs. The response can be an XML document or simply text such as a comma separated response or JSON.

The following is an example of MAKE_REST_REQUEST being used in an application process that is callable by AJAX.

```
declare
  l_clob clob;
  l_buffer         varchar2(32767);
  l_amount         number;
  l_offset         number;
begin

  l_clob := apex_web_service.make_rest_request(
              p_url => 'http://us.music.yahooapis.com/
video/v1/list/published/popular',
              p_http_method => 'GET',
              p_parm_name => apex_util.string_to_table('appid:format'),
              p_parm_value => apex_util.string_to_table(apex_application.g_
x01||':'||apex_application.g_x02));

    l_amount := 32000;
    l_offset := 1;
    begin
        loop
            dbms_lob.read( l_clob, l_amount, l_offset, l_buffer );
            htp.p(l_buffer);
            l_offset := l_offset + l_amount;
            l_amount := 32000;
        end loop;
    exception
        when no_data_found then
            null;
    end;

end;
```

Retrieving Cookies and HTTP Headers

When you invoke a Web service using any of the supported methods in Application Express, the `g_response_cookies` and `g_headers` globals are populated if the Web service response included any cookies or HTTP headers. You can interrogate these globals and store the information in collections.

The following are examples of interrogating the `APEX_WEB_SERVICE` globals to store cookie and HTTP header responses in collections.

```
declare
  i number;
  secure varchar2(1);
begin
  apex_collection.create_or_truncate_collection('P31_RESP_COOKIES');
  for i in 1.. apex_web_service.g_response_cookies.count loop
    IF (apex_web_service.g_response_cookies(i).secure) THEN
      secure := 'Y';
    ELSE
      secure := 'N';
    END IF;
    apex_collection.add_member(p_collection_name => 'P31_RESP_COOKIES',
      p_c001 => apex_web_service.g_response_cookies(i).name,
      p_c002 => apex_web_service.g_response_cookies(i).value,
      p_c003 => apex_web_service.g_response_cookies(i).domain,
      p_c004 => apex_web_service.g_response_cookies(i).expire,
      p_c005 => apex_web_service.g_response_cookies(i).path,
      p_c006 => secure,
      p_c007 => apex_web_service.g_response_cookies(i).version );
  end loop;
end;

declare
  i number;
begin
apex_collection.create_or_truncate_collection('P31_RESP_HEADERS');

for i in 1.. apex_web_service.g_headers.count loop
  apex_collection.add_member(p_collection_name => 'P31_RESP_HEADERS',
    p_c001 => apex_web_service.g_headers(i).name,
    p_c002 => apex_web_service.g_headers(i).value,
    p_c003 => apex_web_service.g_status_code);
end loop;
end;
```

Setting Cookies and HTTP Headers

You set cookies and HTTP headers that should be sent along with a Web service request by populating the globals `g_request_cookies` and `g_request_headers` prior to the process that invokes the Web service.

The following examples show populating the globals to send cookies and HTTP headers with a request.

```
for c1 in (select seq_id, c001, c002, c003, c004, c005, c006, c007
             from apex_collections
            where collection_name = 'P31_RESP_COOKIES' ) loop
  apex_web_service.g_request_cookies(c1.seq_id).name := c1.c001;
  apex_web_service.g_request_cookies(c1.seq_id).value := c1.c002;
  apex_web_service.g_request_cookies(c1.seq_id).domain := c1.c003;
  apex_web_service.g_request_cookies(c1.seq_id).expire := c1.c004;
  apex_web_service.g_request_cookies(c1.seq_id).path := c1.c005;
  if c1.c006 = 'Y' then
    apex_web_service.g_request_cookies(c1.seq_id).secure := true;
  else
    apex_web_service.g_request_cookies(c1.seq_id).secure := false;
  end if;
  apex_web_service.g_request_cookies(c1.seq_id).version := c1.c007;
end loop;

for c1 in (select seq_id, c001, c002
             from apex_collections
            where collection_name = 'P31_RESP_HEADERS' ) loop
  apex_web_service.g_request_headers(c1.seq_id).name := c1.c001;
  apex_web_service.g_request_headers(c1.seq_id).value := c1.c002;
end loop;
```

BLOB2CLOBBASE64 Function

Use this function to convert a BLOB datatype into a CLOB that is base64 encoded. This is often used when sending a binary as an input to a Web service.

Syntax

```
APEX_WEB_SERVICE.BLOB2CLOBBASE64 (
    p_blob IN BLOB)
RETURN CLOB;
```

Parameters

Table 13–1 describes the parameters available in the `BLOB2CLOBBASE64` function.

Table 13–1 BLOB2CLOBBASE64 Parameters

Parameter	Description
`p_blob`	The BLOB to convert into base64 encoded CLOB.

Example

The following example gets a file that was uploaded from the apex_application_files view and converts the BLOB into a CLOB that is base64 encoded.

```
declare
l_clobCLOB;
     l_blobBLOB;
begin
    SELECT BLOB_CONTENT
      INTO l_BLOB
      FROM APEX_APPLICATION_FILES
      WHERE name = :P1_FILE;

    l_CLOB := apex_web_service.blob2clobbase64(l_BLOB);
end;
```

CLOBBASE642BLOB Function

Use this function to convert a CLOB datatype that is base64 encoded into a BLOB. This is often used when receiving output from a Web service that contains a binary parameter.

Syntax

```
APEX_WEB_SERVICE.CLOBBASE642BLOB (
    p_clob IN CLOB)
RETURN BLOB;
```

Parameters

Table 13–2 describes the parameters available in the CLOBBASE642BLOB function.

Table 13–2 CLOBBASE642BLOB Parameters

Parameter	Description
p_clob	The base64 encoded CLOB to convert into a BLOB.

Example

The following example retrieves a base64 encoded node from an XML document as a CLOB and converts it into a BLOB.

```
declare
l_base64CLOB;
     l_blobBLOB;
     l_xml   XMLTYPE;
begin

l_base64 := apex_web_service.parse_xml_clob(l_xml, '
//runReportReturn/reportBytes/text()');

  l_blob := apex_web_service.clobbase642blob(l_base64);
end;
```

MAKE_REQUEST Procedure

Use this procedure to invoke a SOAP style Web service with the supplied SOAP envelope and store the results in a collection.

Syntax

```
APEX_WEB_SERVICE.MAKE_REQUEST (
    p_url               IN VARCHAR2,
    p_action            IN VARCHAR2 default null,
    p_version           IN VARCHAR2 default '1.1',
    p_collection_name   IN VARCHAR2 default null,
    p_envelope          IN CLOB,
    p_username          IN VARCHAR2 default null,
    p_password          IN VARCHAR2 default null,
    p_proxy_override    IN VARCHAR2 default null,
    p_wallet_path       IN VARCHAR2 default null,
    p_wallet_pwd        IN VARCHAR2 default null );
```

Parameters

Table 13–3 describes the parameters available in the MAKE_REQUEST procedure.

Table 13–3 MAKE_REQUEST Procedure Parameters

Parameter	Description
p_url	The URL endpoint of the Web service.
p_action	The SOAP Action corresponding to the operation to be invoked.
p_version	The SOAP version, 1.1 or 1.2. The default is 1.1.
p_collection_name	The name of the collection to store the response.
p_envelope	The SOAP envelope to post to the service.
p_username	The username if basic authentication is required for this service.
p_password	The password if basic authentication is required for this service
p_proxy_override	The proxy to use for the request. The proxy supplied overrides the proxy defined in the application attributes.
p_wallet_path	The file system path to a wallet if the URL endpoint is https. For example, file:/usr/home/oracle/WALLETS. The wallet path provided overrides the wallet defined in the instance settings.
p_wallet_pwd	The password to access the wallet.

Example

The following example uses the make_request procedure to retrieve a list of movies from a SOAP style Web service. The response is stored in an Application Express collection named MOVIE_LISTINGS.

```
declare
 l_envelope CLOB;
BEGIN
l_envelope := '<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="http://www.ignyte.com/whatsshowing"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
   <soap:Body>
      <tns:GetTheatersAndMovies>
         <tns:zipCode>43221</tns:zipCode>
         <tns:radius>5</tns:radius>
      </tns:GetTheatersAndMovies>
   </soap:Body>
</soap:Envelope>';

apex_web_service.make_request(
   p_url               => '
http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/moviefunctions.as
mx',
   p_action            => '
http://www.ignyte.com/whatsshowing/GetTheatersAndMovies',
   p_collection_name   => 'MOVIE_LISTINGS',
   p_envelope          => l_envelope
);
END;
```

MAKE_REQUEST Function

Use this function to invoke a SOAP style Web service with the supplied SOAP envelope returning the results in an XMLTYPE.

Syntax

```
APEX_WEB_SERVICE.MAKE_REQUEST (
    p_url               IN VARCHAR2,
    p_action            IN VARCHAR2 default null,
    p_version           IN VARCHAR2 default '1.1',
    p_envelope          IN CLOB,
    p_username          IN VARCHAR2 default null,
    p_password          IN VARCHAR2 default null,
    p_proxy_override    IN VARCHAR2 default null,
    p_wallet_path       IN VARCHAR2 default null,
    p_wallet_pwd        IN VARCHAR2 default null )
RETURN XMLTYPE;
```

Parameters

Table 13–4 describes the parameters available in the `MAKE_REQUEST` function.

Table 13–4 MAKE_REQUEST Function Parameters

Parameter	Description
`p_url`	The URL endpoint of the Web service.
`p_action`	The SOAP Action corresponding to the operation to be invoked.
`p_version`	The SOAP version, 1.1 or 1.2. The default is 1.1.
`p_envelope`	The SOAP envelope to post to the service.
`p_username`	The username if basic authentication is required for this service.
`p_password`	The password if basic authentication is required for this service
`p_proxy_override`	The proxy to use for the request. The proxy supplied overrides the proxy defined in the application attributes.
`p_wallet_path`	The file system path to a wallet if the URL endpoint is https. For example, file:/usr/home/oracle/WALLETS. The wallet path provided overrides the wallet defined in the instance settings.
`p_wallet_pwd`	The password to access the wallet.

Example

The following example uses the `make_request` function to invoke a SOAP style Web service that returns movie listings. The result is stored in an XMLTYPE.

```
declare
    l_envelope CLOB;
    l_xmlXMLTYPE;
BEGIN
    l_envelope := ' <?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="http://www.ignyte.com/whatsshowing"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <soap:Body>
        <tns:GetTheatersAndMovies>
```

```
            <tns:zipCode>43221</tns:zipCode>
            <tns:radius>5</tns:radius>
        </tns:GetTheatersAndMovies>
    </soap:Body>
</soap:Envelope>';

l_xml := apex_web_service.make_request(
    p_url => '
http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/moviefunctions.as
mx',
   p_action => ' http://www.ignyte.com/whatsshowing/GetTheatersAndMovies',
   p_envelope => l_envelope
);
END
```

MAKE_REST_REQUEST Function

Use this function to invoke a RESTful style Web service supplying either name value pairs, a character based payload or a binary payload and returning the response in a CLOB.

Syntax

```
APEX_WEB_SERVICE.MAKE_REST_REQUEST(
    p_url               IN VARCHAR2,
    p_http_method       IN VARCHAR2,
    p_username          IN VARCHAR2 default null,
    p_password          IN VARCHAR2 default null,
    p_proxy_override    IN VARCHAR2 default null,
    p_body              IN CLOB default empty_clob(),
    p_body_blob         IN BLOB default empty_blob(),
    p_parm_name         IN WWV_FLOW_GLOBAL.VC_ARR2 default empty_vc_arr,
    p_parm_value        IN WWV_FLOW_GLOBAL.VC_ARR2 default empty_vc_arr,
    p_wallet_path       IN VARCHAR2 default null,
    p_wallet_pwd        IN VARCHAR2 default null )
RETURN CLOB;
```

Parameters

Table 13–5 describes the parameters available in the `MAKE_REST_REQUEST` function.

Table 13–5 MAKE_REST_REQUEST Function Parameters

Parameter	Description
`p_url`	The URL endpoint of the Web service.
`p_http_method`	The HTTP method to use, PUT, POST, GET, HEAD, or DELETE.
`p_username`	The username if basic authentication is required for this service.
`p_password`	The password if basic authentication is required for this service
`p_proxy_override`	The proxy to use for the request. The proxy supplied overrides the proxy defined in the application attributes.
`p_body`	The HTTP payload to be sent as CLOB.
`p_body_blob`	The HTTP payload to be sent as binary BLOB. For example, posting a file.
`p_parm_name`	The name of the parameters to be used in name/value pairs.
`p_parm_value`	The value of the parameters to be used in name/value pairs.
`p_wallet_path`	The file system path to a wallet if the URL endpoint is https. For example, file:/usr/home/oracle/WALLETS. The wallet path provided overrides the wallet defined in the instance settings.
`p_wallet_pwd`	The password to access the wallet.

Example

The following example calls a RESTful style Web service using the `make_rest_request` function passing the parameters to the service as name/value pairs. The response from the service is stored in a locally declared CLOB.

```
declare
    l_clob CLOB;
```

```
BEGIN

    l_clob := apex_web_service.make_rest_request(
        p_url => 'http://us.music.yahooapis.com/ video/v1/list/published/popular',
        p_http_method => 'GET',
        p_parm_name => apex_util.string_to_table('appid:format'),
        p_parm_value => apex_util.string_to_table('xyz:xml'));

END
```

PARSE_RESPONSE Function

Use this function to parse the response from a Web service that is stored in a collection and return the result as a VARCHAR2 type.

Syntax

```
APEX_WEB_SERVICE.PARSE_RESPONSE (
    p_collection_name   IN VARCHAR2,
    p_xpath             IN VARCHAR2,
    p_ns                IN VARCHAR2 default null )
RETURN VARCHAR2;
```

Parameters

Table 13–6 describes the parameters available in the PARSE_RESPONSE function.

Table 13–6 PARSE_RESPONSE Function Parameters

Parameter	Description
p_collection_name	The name of the collection where the Web service response is stored.
p_xpath	The XPath expression to the desired node.
p_ns	The namespace to the desired node.

Example

The following example parses a response stored in a collection called STELLENT_CHECKIN and stores the value in a locally declared VARCHAR2 variable.

```
declare
    l_response_msg  VARCHAR2(4000);
BEGIN
    l_response_msg := apex_web_service.parse_response(
        p_collection_name=>'STELLENT_CHECKIN',
        p_xpath =>
'//idc:CheckInUniversalResponse/idc:CheckInUniversalResult/idc:StatusInfo/idc:stat
usMessage/text()',
        p_ns=>'xmlns:idc="http://www.stellent.com/CheckIn/"');
END;
```

PARSE_RESPONSE_CLOB Function

Use this function to parse the response from a Web service that is stored in a collection and return the result as a CLOB type.

Syntax

```
APEX_WEB_SERVICE.PARSE_RESPONSE (
    p_collection_name   IN VARCHAR2,
    p_xpath             IN VARCHAR2,
    p_ns                IN VARCHAR2 default null )
RETURN CLOB;
```

Parameters

Table 13–7 describes the parameters available in the PARSE_RESPONSE_CLOB function.

Table 13–7 PARSE_RESPONSE _CLOB Function Parameters

Parameter	Description
p_collection_name	The name of the collection where the Web service response is stored.
p_xpath	The XPath expression to the desired node.
p_ns	The namespace to the desired node.

Example

The following example parses a response stored in a collection called STELLENT_CHECKIN and stores the value in a locally declared CLOB variable.

```
declare
    l_response_msg  CLOB;
BEGIN
    l_response_msg := apex_web_service.parse_response(
        p_collection_name=>'STELLENT_CHECKIN',
        p_xpath=>
'//idc:CheckInUniversalResponse/idc:CheckInUniversalResult/idc:StatusInfo/idc:stat
usMessage/text()',
        p_ns=>'xmlns:idc="http://www.stellent.com/CheckIn/"');
END;
```

PARSE_XML Function

Use this function to parse the response from a Web service returned as an XMLTYPE and return the value requested as a VARCHAR2.

Syntax

```
APEX_WEB_SERVICE.PARSE_XML (
    p_xml               IN XMLTYPE,
    p_xpath             IN VARCHAR2,
    p_ns                IN VARCHAR2 default null )
RETURN VARCHAR2;
```

Parameters

Table 13–8 describes the parameters available in the `PARSE_XML` function.

Table 13–8 PARSE_XML Function Parameters

Parameter	Description
`p_xml`	The XML document as an XMLTYPE to parse.
`p_xpath`	The XPath expression to the desired node.
`p_ns`	The namespace to the desired node.

Example

The following example uses the `make_request` function to call a Web service and store the results in a local XMLTYPE variable. The `parse_xml` function is then used to pull out a specific node of the XML document stored in the XMLTYPE and stores it in a locally declared VARCHAR2 variable.

```
declare
    l_envelope CLOB;
    l_xml XMLTYPE;
    l_movie VARCHAR2(4000);
BEGIN
    l_envelope := ' <?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="http://www.ignyte.com/whatsshowing"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
      <tns:GetTheatersAndMovies>
         <tns:zipCode>43221</tns:zipCode>
         <tns:radius>5</tns:radius>
      </tns:GetTheatersAndMovies>
   </soap:Body>
</soap:Envelope>';

   l_xml := apex_web_service.make_request(
     p_url => '
http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/moviefunctions.as
mx',
     p_action => ' http://www.ignyte.com/whatsshowing/GetTheatersAndMovies',
     p_envelope => l_envelope );

   l_movie := apex_web_service.parse_xml(
     p_xml => l_xml,
```

```
     p_xpath => '
//GetTheatersAndMoviesResponse/GetTheatersAndMoviesResult/Theater/Movies/Movie/Nam
e[1]',
     p_ns => ' xmlns="http://www.ignyte.com/whatsshowing"' );

END;
```

PARSE_XML_CLOB Function

Use this function to parse the response from a Web service returned as an XMLTYPE and return the value requested as a CLOB.

Syntax

```
APEX_WEB_SERVICE.PARSE_XML_CLOB (
    p_xml               IN XMLTYPE,
    p_xpath             IN VARCHAR2,
    p_ns                IN VARCHAR2 default null )
RETURN VARCHAR2;
```

Parameters

Table 13–9 describes the parameters available in the `PARSE_XML_CLOB` function.

Table 13–9 PARSE_XML_CLOB Function Parameters

Parameter	Description
`p_xml`	The XML document as an XMLTYPE to parse.
`p_xpath`	The XPath expression to the desired node.
`p_ns`	The namespace to the desired node.

Example

The following example uses the `make_request` function to call a Web service and store the results in a local XMLTYPE variable. The `parse_xml` function is then used to pull out a specific node of the XML document stored in the XMLTYPE and stores it in a locally declared VARCHAR2 variable

```
declare
    l_envelope CLOB;
    l_xml XMLTYPE;
    l_movie CLOB;
BEGIN
    l_envelope := ' <?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="http://www.ignyte.com/whatsshowing"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
      <tns:GetTheatersAndMovies>
         <tns:zipCode>43221</tns:zipCode>
         <tns:radius>5</tns:radius>
      </tns:GetTheatersAndMovies>
   </soap:Body>
</soap:Envelope>';

   l_xml := apex_web_service.make_request(
     p_url => '
http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/moviefunctions.as
mx',
     p_action => ' http://www.ignyte.com/whatsshowing/GetTheatersAndMovies',
     p_envelope => l_envelope );

   l_movie := apex_web_service.parse_xml(
     p_xml => l_xml,
```

```
    p_xpath => '
//GetTheatersAndMoviesResponse/GetTheatersAndMoviesResult/Theater/Movies/Movie/Nam
e[1]',
    p_ns => ' xmlns="http://www.ignyte.com/whatsshowing"' );

END;
```

Index

A

APEX_APPLICATION
  global variables, 4-1
  HELP Procedure, 4-6
  package, 4-1
  Referencing Arrays
    referencing, 4-3
APEX_COLLECTION, 12-3
  ADD_MEMBER function, 12-14
  ADD_MEMBER procedure, 12-12
  ADD_MEMBERS procedure, 12-16
  COLLECTION_EXISTS function, 12-18
  COLLECTION_HAS_CHANGED function, 12-19
  COLLECTION_MEMBER_COUNT function, 12-20
  CREATE_COLLECTION procedure, 12-21
  CREATE_COLLECTION_FROM_QUERY, 12-23
  CREATE_COLLECTION_FROM_QUERY_B procedure, 12-27
  CREATE_COLLECTION_FROM_QUERY2 procedure, 12-25
  CREATE_COLLECTION_FROM_QUERYB2 procedure, 12-29
  CREATE_OR_TRUNCATE_ COLLECTION, 12-11
  CREATE_OR_TRUNCATE_COLLECTION procedure, 12-22
  DELETE_ALL_COLLECTIONS procedure, 12-31
  DELETE_ALL_COLLECTIONS_SESSION procedure, 12-32
  DELETE_COLLECTION procedure, 12-33
  DELETE_MEMBER procedure, 12-34
  DELETE_MEMBERS procedure, 12-35
  GET_MEMBER_MD5 function, 12-36
  MERGE_MEMBERS procedure, 12-37
  MOVE_MEMBER_DOWN procedure, 12-39
  MOVE_MEMBER_UP procedure, 12-40
  RESEQUENCE_COLLECTION procedure, 12-41
  RESET_COLLECTION_CHANGED procedure, 12-42
  RESET_COLLECTION_CHANGED_ALL procedure, 12-43
  SORT_MEMBERS procedure, 12-44
  TRUNCATE_COLLECTION procedure, 12-45
  UPDATE_MEMBER procedure, 12-46
  UPDATE_MEMBER_ATTRIBUTE procedure signature 1, 12-50
  UPDATE_MEMBER_ATTRIBUTE procedure signature 2, 12-52
  UPDATE_MEMBER_ATTRIBUTE procedure signature 3, 12-54
  UPDATE_MEMBER_ATTRIBUTE procedure signature 4, 12-56
  UPDATE_MEMBER_ATTRIBUTE procedure signature 5, 12-58
  UPDATE_MEMBER_ATTRIBUTE procedure signature 6, 12-60
  UPDATE_MEMBERS procedure, 12-48
APEX_CUSTOM_AUTH, 5-1
  APPLICATION_PAGE_ITEM_EXISTS function, 5-2
  CURRENT_PAGE_IS_PUBLIC function, 5-3
  DEFINE_USER_SESSION procedure, 5-4
  GET_COOKIE_PROPS, 5-5
  GET_LDAP_PROPS, 5-6
  GET_NEXT_SESSION_ID function, 5-8
  GET_SECURITY_GROUP_ID function, 5-9
  GET_SESSION_ID function, 5-10
  GET_SESSION_ID_FROM_COOKIE, 5-11
  GET_USER function, 5-12
  GET_USERNAME, 5-13
  IS_SESSION_VALID, 5-14
  LOGIN
    Login API, 5-15
  LOGOUT, 5-16
  POST_LOGIN, 5-17
  SESSION_ID_EXISTS function, 5-18
  SET_SESSION_ID procedure, 5-19
  SET_SESSION_ID_TO_NEXT_VALUE procedure, 5-20
  SET_USER procedure, 5-21
APEX_INSTANCE_ADMIN, 7-1
  ADD_SCHEMA procedure, 7-2
  ADD_WORKSPACE procedure, 7-3
  GET_PARAMETER function, 7-4
  GET_SCHEMAS function, 7-5
  parameter values, 7-11
  REMOVE_SAVED_REPORT procedure, 7-7
  REMOVE_SAVED_REPORTS procedure, 7-6
  REMOVE_SCHEMA procedure, 7-8
  REMOVE_WORKSPACE procedure, 7-9

SET_PARAMETER procedure, 7-10
APEX_ITEM, 3-1
CHECKBOX function, 3-2
DATE_POPUP function, 3-4
DISPLAY_AND_SAVE, 3-6
HIDDEN function, 3-7
MD5_CHECKSUM function, 3-9
MD5_HIDDEN function, 3-10
POPUP_FROM_LOV function, 3-11
POPUP_FROM_QUERY function, 3-13
POPUPKEY_FROM_LOV function, 3-15
POPUPKEY_FROM_QUERY function, 3-17
RADIOGROUP function, 3-19
SELECT_LIST function, 3-20
SELECT_LIST_FROM_LOV function, 3-22
SELECT_LIST_FROM_LOV_XL function, 3-23
SELECT_LIST_FROM_QUERY function, 3-25
SELECT_LIST_FROM_QUERY_XL function, 3-27
TEXT function, 3-29
TEXT_FROM_LOV function, 3-31
TEXT_FROM_LOV_QUERY function, 3-32
TEXTAREA function, 3-30
APEX_LANG, 11-1
LANG function, 11-2
MESSAGE function, 11-3
APEX_LDAP, 6-1
AUTHENTICATE, 6-2
GET_ALL_USER_ATTRIBUTES, 6-3
GET_USER_ATTRIBUTES, 6-5
IS_MEMBER, 6-7
MEMBER_OF, 6-9
MEMBER_OF2 Function, 6-10
APEX_MAIL, 2-1
ADD_ATTACHMENT procedure, 2-3
PUSH_QUEUE procedure, 2-4
SEND procedure, 2-5
APEX_MAIL_QUEUE
sending email in queue, 2-4
APEX_PLSQL_JOB, 10-1
APEX_PLSQL_JOBS
JOBS_ARE_ENABLED Function, 10-3
PURGE_PROCESS Procedure, 10-4
SUBMIT_PROCESS Function, 10-5
TIME_ELAPSED Function, 10-6
UPDATE_JOB_STATUS Procedure, 10-7
APEX_UI_DEFAULT_UPDATE, 8-1
ADD_AD_COLUMN procedure, 8-3
ADD_AD_SYNONYM procedure, 8-5
DEL_AD_COLUMN procedure, 8-6
DEL_AD_SYNONYM procedure, 8-7
DEL_COLUMN procedure, 8-8
DEL_GROUP procedure, 8-9
DEL_TABLE procedure, 8-10
SYNCH_TABLE procedure, 8-11
UPD_AD_COLUMN procedure, 8-12
UPD_AD_SYNONYM procedure, 8-14
UPD_COLUMN procedure, 8-15
UPD_DISPLAY_IN_FORM procedure, 8-17
UPD_DISPLAY_IN_REPORT procedure, 8-18
UPD_FORM_REGION_TITLE procedure, 8-19
UPD_GROUP procedure, 8-20
UPD_ITEM_DISPLAY_HEIGHT procedure, 8-21
UPD_ITEM_DISPLAY_WIDTH procedure, 8-22
UPD_ITEM_FORMAT_MASK procedure, 8-23
UPD_ITEM_HELP procedure, 8-24
UPD_ITEM_LABEL procedure, 8-25
UPD_REPORT_ALIGNMENT procedure, 8-26
UPD_REPORT_FORMAT_MASK procedure, 8-27
UPD_REPORT_REGION_TITLE procedure, 8-28
UPD_TABLE procedure, 8-29
APEX_UTIL, 1-1
CACHE_GET_DATE_OF_PAGE_CACHE function, 1-5, 1-6
CACHE_PURGE_BY_APPLICATION procedure, 1-7
CACHE_PURGE_BY_PAGE procedure, 1-8
CACHE_PURGE_STALE procedure, 1-9
CHANGE_CURRENT_USER_PW procedure, 1-10
CHANGE_PASSWORD_ON_FIRST_USE function, 1-11
CLEAR_APP_CACHE procedure, 1-12
CLEAR_PAGE_CACHE procedure, 1-13
CLEAR_USER_CACHE procedure, 1-14
COUNT_CLICK procedure, 1-15
CREATE_USER procedure, 1-16
CREATE_USER_GROUP procedure, 1-19
CURRENT_USER_IN_GROUP function, 1-20
DOWNLOAD_PRINT_DOCUMENT procedure signature 1, 1-21
DOWNLOAD_PRINT_DOCUMENT procedure signature 2, 1-22
DOWNLOAD_PRINT_DOCUMENT procedure signature 3, 1-24
DOWNLOAD_PRINT_DOCUMENT procedure signature 4, 1-26
EDIT_USER procedure, 1-27
END_USER_ACCOUNT_DAYS_LEFT function, 1-31
EXPIRE_END_USER_ACCOUNT procedure, 1-32
EXPIRE_WORKSPACE_ACCOUNT procedure, 1-33
EXPORT_USERS procedure, 1-34
FETCH_APP_ITEM function, 1-35
FETCH_USER procedure signature 1, 1-36
FETCH_USER procedure signature 2, 1-39
FETCH_USER procedure signature 3, 1-41
FIND_SECURITY_GROUP_ID function, 1-44
FIND_WORKSPACE function, 1-45
GET_ACCOUNT_LOCKED_STATUS function, 1-46
GET_ATTRIBUTE function, 1-47
GET_AUTHENTICATION_RESULT function, 1-48
GET_BLOB_FILE_SRC function, 1-49
GET_CURRENT_USER_ID function, 1-51
GET_DEFAULT_SCHEMA function, 1-52
GET_EDITION function, 1-53

GET_EMAIL function, 1-54
GET_FEEDBACK_FOLLOW_UP function, 1-55
GET_FILE procedure, 1-56
GET_FILE_ID function, 1-58
GET_FIRST_NAME function, 1-59
GET_GROUP_ID function, 1-61
GET_GROUP_NAME function, 1-62
GET_GROUPS_USER_BELONGS_TO function, 1-60
GET_LAST_NAME function, 1-63
GET_NUMERIC_SESSION_STATE function, 1-64
GET_PREFERENCE function, 1-65
GET_PRINT_DOCUMENT function signature 1, 1-66
GET_PRINT_DOCUMENT function signature 2, 1-67
GET_PRINT_DOCUMENT function signature 3, 1-68
GET_PRINT_DOCUMENT function signature 4, 1-69
GET_SCREEN_READER_MODE_TOGGLE function, 1-71
GET_SESSION_LANG function, 1-72
GET_SESSION_STATE function, 1-73
GET_SESSION_TERRITORY function, 1-74
GET_SESSION_TIME_ZONEfunction, 1-75
GET_USER_ID function, 1-76
GET_USER_ROLES function, 1-77
GET_USERNAME function, 1-78
IR_CLEAR procedure, 1-79
IR_FILTER procedure, 1-80
IR_RESET procedure, 1-81
IS_LOGIN_PASSWORD_VALID function, 1-82
IS_SCREEN_READER_SESSION function, 1-83
IS_SCREEN_READER_SESSION_YN function, 1-84
IS_USERNAME_UNIQUE function, 1-85
KEYVAL_NUM function, 1-86
KEYVAL_VC2 function, 1-87
LOCK_ACCOUNT procedure, 1-88, 1-136
PASSWORD_FIRST_USE_OCCURRED function, 1-89
PREPARE_URL function, 1-90
PUBLIC_CHECK_AUTHORIZATION function, 1-92
PURGE_REGIONS_BY_APP procedure, 1-93
PURGE_REGIONS_BY_NAME procedure, 1-94
PURGE_REGIONS_BY_PAGE procedure, 1-95
REMOVE_PREFERENCE procedure, 1-96
REMOVE_SORT_PREFERENCES procedure, 1-97
REMOVE_USER procedure, 1-98
RESET_AUTHORIZATIONS procedure, 1-99
RESET_PW procedure, 1-100
SAVEKEY_NUM function, 1-101
SAVEKEY_VC2 function, 1-102
SET_ATTRIBUTE procedure, 1-103
SET_AUTHENTICATION_RESULT procedure, 1-104
SET_CUSTOM_AUTH_STATUS procedure, 1-105
SET_EDITION procedure, 1-106
SET_EMAIL procedure, 1-107
SET_FIRST_NAME procedure, 1-108
SET_LAST_NAME procedure, 1-109
SET_PREFERENCE procedure, 1-110
SET_SECURITY_GROUP_ID procedure, 1-111, 1-112
SET_SESSION_SCREEN_READER_OFF procedure, 1-117, 1-118
SET_SESSION_STATE procedure, 1-113, 1-115, 1-119
SET_SESSION_TERRITORY procedure, 1-120
SET_SESSION_TIME_ZONE procedure, 1-121
SET_USERNAME procedure, 1-122
SHOW_SCREEN_READER_MODE_TOGGLE function, 1-123
STRING_TO_TABLE function, 1-124
STRONG_PASSWORD_CHECK procedure, 1-125
STRONG_PASSWORD_VALIDATION function, 1-129
SUBMIT_FEEDBACK procedure, 1-130
SUBMIT_FEEDBACK_FOLLOWUP procedure, 1-132
TABLE_TO_STRING function, 1-133
UNEXPIRE_END_USER_ACCOUNT procedure, 1-134
UNEXPIRE_WORKSPACE_ACCOUNT procedure, 1-135
URL_ENCODE function, 1-137
WORKSPACE_ACCOUNT_DAYS_LEFT function, 1-139
APEX_WEB_SERVICE, 13-1
About the APEX_WEB_SERVICE API, 13-2
BLOB2CLOBBASE64 function, 13-8
CLOBBASE642BLOB function, 13-9
Invoking a RESTful Style Web Service, 13-5
Invoking a SOAP Style Web Service, 13-3
MAKE_REQUEST function, 13-12
MAKE_REQUEST procedure, 13-10
MAKE_REST_REQUEST function, 13-14
PARSE_RESPONSE function, 13-16
PARSE_RESPONSE_CLOB function, 13-17
PARSE_XML function, 13-18
PARSE_XML_CLOB function, 13-20
Retrieving Cookies and HTTP HeadersI, 13-6
Setting Cookies and HTTP Headers, 13-7
APIs
APEX_APPLICATION, 4-1
APEX_COLLECTION, 12-3
APEX_CUSTOM_AUTH, 5-1
APEX_INSTANCE_ADMIN, 7-1
APEX_ITEM, 3-1
APEX_LANG, 11-1
APEX_LDAP, 6-1
APEX_MAIL, 2-1
APEX_PLSQL_JOB, 10-1
APEX_UI_DEFAULT_UPDATE, 8-1
APEX_UTIL, 1-1

APEX_WEB_SERVICE, 13-1
JavaScript API, 9-1
application
sending messages in APEX_MAIL_QUEUE, 2-4
sending outbound email, 2-5
sending outbound email as attachment, 2-3
attribute values
setting, 1-103
authenticated user
create user group, 1-19
authentication
scheme session cookies, 5-5

C

check box
creating, 3-2
clicks
counting, 1-15
collections
accessing, 12-5
APEX_COLLECTION API, 12-3
clearing session state, 12-10
creating, 12-4
deleting members, 12-9
determining status, 12-11
managing, 12-10
merging, 12-7
truncating, 12-7
updating members, 12-8

E

email
sending as an attachment, 2-3
sending messages in APEX_MAIL_QUEUE, 2-4
sending outbound, 2-5
export file
of workspace, 1-34

F

F01, 4-3
file repository
downloading files, 1-56
obtaining primary key, 1-58
Function, 13-20

J

JavaScript API, 9-1
$d_ClearAndHide(pNd), 9-40
$d_Find(pThis,pString,pTags,pClass), 9-76
$dom_
AddInput(pThis,pType,pId,pName,pValue), 9-71
$dom_AddTag(pThis, pTag, pText), 9-68
$dom_MakeParent(p_Node,p_Parent), 9-72
$f_CheckAll(pThis, pCheck, pArray), 9-54
$f_CheckFirstColumn(pNd), 9-55
$f_DisableOnValue(pThis, pValue, pThat), 9-50
$f_First_field(pNd), 9-79
$f_get_emptys(pNd, pClassFail, pClass), 9-37
$f_Hide_On_Value_Item(pThis, pThat, pValue), 9-46
$f_Hide_On_Value_Item_Row(pThis, pThat, pValue), 9-48
$f_ReturnChecked(pNd), 9-39
$f_SelectedOptions(pNd), 9-41
$f_SelectValue(pNd), 9-42
$f_SetValueSequence(pArray,pMultiple), 9-67
$f_Show_On_Value_Item(pThis, pThat, pValue), 9-47
$f_Show_On_Value_Item_Row(pThis, pThat, pValue), 9-49
$f_Swap(pThis,pThat), 9-65
$f_ValuesToArray(pThis, pClass, pTag), 9-52
$nvl(pTest, pDefault), 9-12
$s(pNd, pValue), 9-9
$tr_AddTD(pThis,pText), 9-69
$tr_AddTH(pThis,pText), 9-70
$u_ArrayToString(pArray, pDelim), 9-43
$u_Carray(pNd), 9-10
$u_Narray(pNd), 9-11
$u_SubString(pText,pMatch), 9-60
$v(pNd), 9-7
$v_Array(pNd), 9-38
$v_CheckValueAgainst(pThis, pValue), 9-45
$v_IsEmpty(pThis), 9-62
$v_PopupReturn(pValue, pThat), 9-56
$v_Upper(pNd), 9-75
$v2(pNd), 9-8
$x(pNd), 9-6
$x_ByClass(pClass, pNd, pTag), 9-32
$x_CheckImageSrc(pId,pSearch), 9-44
$x_CheckImageSrc(pNd, pSearch), 9-59
$x_Class(pNd,pClass), 9-30
$x_ClassByClass(pNd, pClass, pTag, pClass2), 9-51
$x_disableItem(pNd,a), 9-36
$x_FormItems(pNd, pType), 9-53
$x_Hide(pNd), 9-17
$x_HideAllExcept(pNd,pNdArray), 9-27
$x_HideItemRow(pNd), 9-24
$x_HideSiblings(pNd), 9-28
$x_ItemRow(pNd,pFunc), 9-23
$x_Remove(pNd), 9-20
$x_RowHighlight(pThis,pColor), 9-73
$x_RowHighlightOff(pThis), 9-74
$x_SetSiblingsClass(pNd, pClass, pNdClass), 9-31
$x_Show(pNd), 9-18
$x_ShowAllByClass(pNd, pClass, pTag), 9-33
$x_ShowChildren(pThis), 9-34, 9-35
$x_ShowItemRow(pNd), 9-25
$x_ShowSiblings(pNd), 9-29
$x_Style(pNd, pStyle, pString), 9-16
$x_SwitchImageSrc(pNd, pSearch, pReplace), 9-58
$x_Toggle(pNd), 9-19
$x_ToggleItemRow(pNd), 9-26

$x_ToggleWithImage(pThis,pNd), 9-57
$x_UpTill(pNd, pToTag), 9-22
$x_Value(pNd,pValue), 9-21
addLoadEvent(pFunction), 9-64
apex.event.trigger(pSelector,pEvent,dData), 9-4
apex.widget.initPageItem(pName,pOptions), 9-5
confirmDelete(pMessage, pRequest), 9-14, 9-15
doSubmit(pRequest), 9-13
GetCookie (pName), 9-80
html_RemoveAllChildren(pNd), 9-61
html_SetSelectValue(pId,pValue), 9-63
returnInput(p_R, p_D), 9-77
SetCookie (pName,pValue), 9-81
setReturn(p_R,p_D), 9-78
submitEnter(pNd,e), 9-66

L

LDAP attributes
obtaining, 5-6

P

password
changing, 1-10
resetting and emailing, 1-100

R

radio group
generate, 3-19

S

s, 12-7
session
cookies, 5-5
session state
fetching for current application, 1-35
removing for current page, 1-13
removing for current session, 1-12
setting, 1-113, 1-115, 1-119
special characters
encoding, 1-137

U

user
get e-mail address, 1-54
remove preference, 1-96
user account
altering, 1-27
creating new, 1-16
fetching, 1-36, 1-39, 1-41
removing, 1-98
update email address, 1-107
updating FIRST_NAME, 1-108
updating LAST_NAME value, 1-109
updating USER_NAME value, 1-122

V

variables
global, 4-1

W

workspace
export file, 1-34
numeric security group ID, 1-44